Detection of cells' nuclei using DETR

Ronit Feldman
308106988, ronitfeldman@mail.tau.ac.il

Abstract

Object detection is a computer vision technique for detecting the objects contained in an image or video with the aid of Image classification, redrawing a limiting box around each object, and assigning a class label obtained from the image. We often encounter detection tasks in biomedical applications. Examples range from detecting lesions on functional magnetic resonance images, to the detection of tumors in histopathological images. We will show a comparison of the main detection method: FAST C-RNN[12], YOLO[15], DETR[11] on nuclear cell data. Attention-based transformers are state-of-the-art in a range of deep learning fields. They have recently been proposed for detection tasks where they are beginning to outperforming other methods. There was only one article that used DETR (DEtection TRansformer) on biological data[11]. We will recreate and comer their results on a different data set. We showcase the method's contribution in a typical use case of detecting nuclei in cell environments, commonly employed in medical researchers. For the specific use case, the proposed method was only slightly worse than other state-of-the-art tools for image detection. The fast and accurate detection performance enables a posterior data processing so that experiments can be monitored in real time and closed-loop optimization can be performed. Code and data samples are available at https://github.com/ronitfe/Detection-of-cells-nuclei-using-DETR.git.

Keywords: attention, detection, transformers, single-cell analysis, deep learning.

1 Introduction

Imagine accelerating research to find treatments for almost every disease, from lung cancer and heart disease to rare disorders. Many lives would be reshaped if cures came sooner. Possibly, we could find cures for the world's illnesses, from the common cold to rare disorders, by automating nucleus detection. Identifying the cells' nuclei is the starting point for most analyses because most of the human body's 30 trillion cells contain a nucleus full of DNA, the genetic code that programs each cell. Identifying nuclei allows researchers to

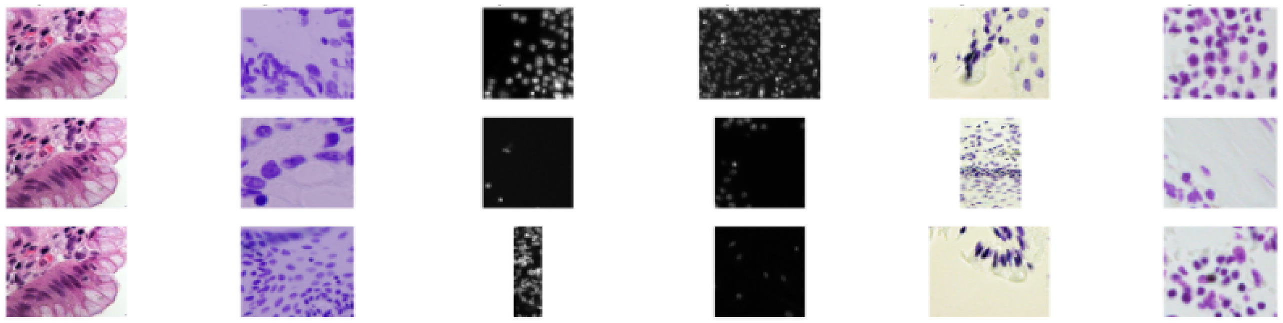

Figure 1: example of data-set images

identify each cell in a sample, and by measuring how cells react to various treatments, the researcher can understand the underlying biological processes at work.

Our purpose is to create a bounding box on nuclei's in cells.

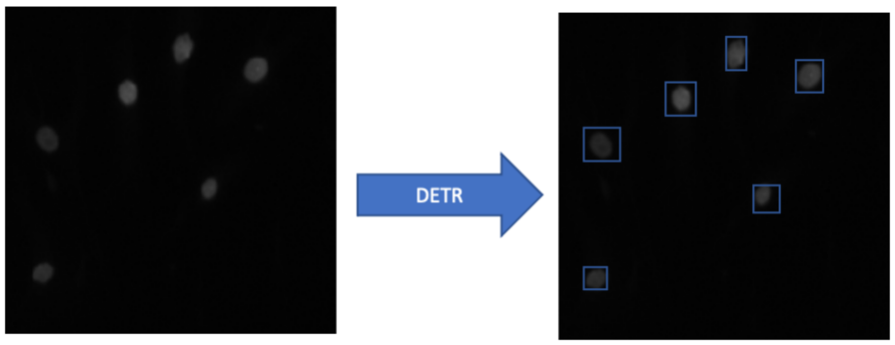

Figure 2: detection bounding box example

We successfully detected nuclei in a variety of images. A major advantage of the model is its speed, simplicity, and ease of maintenance.

2 Related Work

Computer vision is an interdisciplinary field that has been gaining huge amounts of traction in recent years(since CNN). one of the integral parts of computer vision is object detection. Object detection aids in pose estimation, vehicle detection, surveillance, etc. The difference between object detection algorithms and classification algorithms is that in detection algorithms, we try to draw a bounding box around the object of interest to locate it within the image. The major reason why you cannot proceed with this problem by building a standard convolutional network followed by a fully connected layer is that the length of the output layer is variable — not constant, this is because the number of occurrences of the objects of interest is not fixed. [7] A naive approach to solve this problem would be to take different regions of interest from the image and use a CNN to classify the presence of the object within that region. There are numerous methods for classic image detection using binding boxes. We would not elaborate on the time-consuming method but compare it to those who can give online results like DETR.

2.1 *Fast R-CNN*

We will first talk about R-CNN[13]. To bypass the problem of selecting thousands of regions,a method of detecting just 2000 regions in an image was proposed. first Gen-

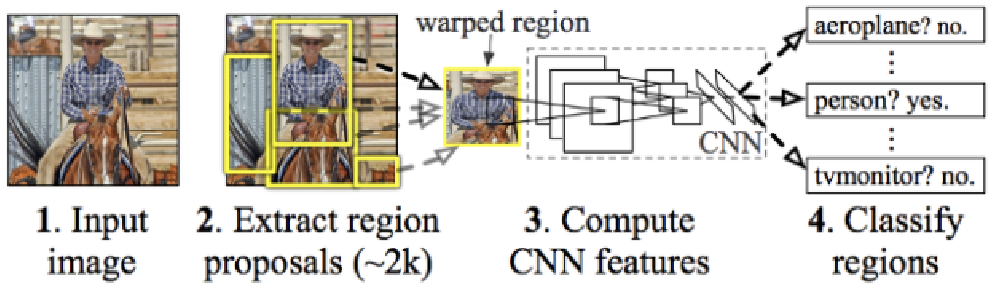

Figure 3: R-CNN architecture

erate initial sub-segmentation, we generate many candidate regions second, Use a greedy algorithm to recursively combine similar regions into larger ones then Use the generated regions to produce the final candidate region proposals But the problem is It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image. It cannot be implemented in real-time.

The same author of the previous paper(R-CNN) invented a new approach that is similar to the R-CNN algorithm, however, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares, and by using an RoI pooling layer we reshape them into a fixed size so that they can be fed into a fully connected layer. From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box. We don't have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it, however, The network does not look at the complete image.

2.2 *YOLO*

The boundaries of a box and its class probabilities are determined by a single convolutional network in YOLO[15]. we take an image and split it into an SxS grid, within each of the

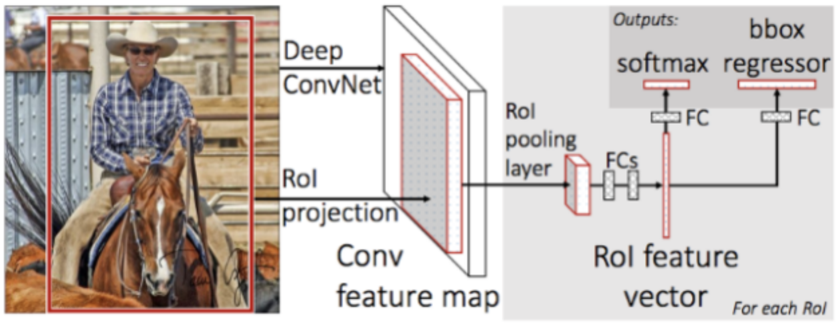

Figure 4: Fast R-CNN architecture

grid we take m bounding boxes. For each of the bounding boxes, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value are selected and used to locate the object within the image. [**?**].

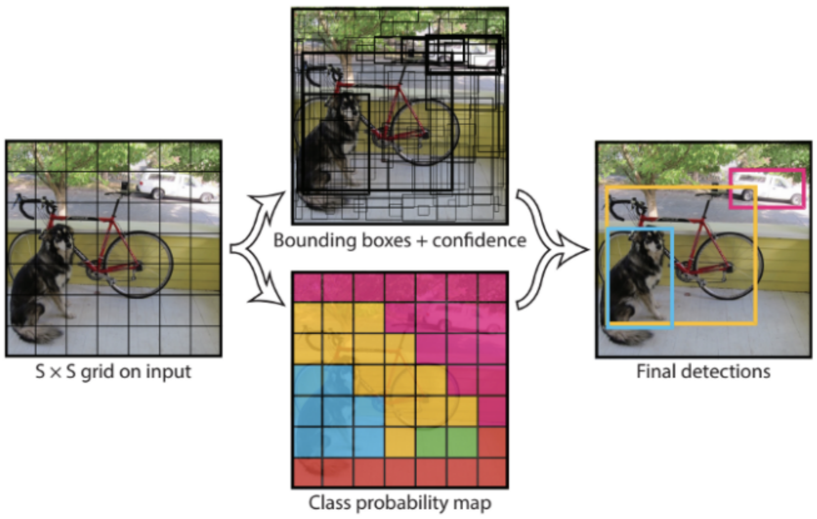

Figure 5: YOLO architecture

Due to the spatial constraints of the YOLO algorithm, the algorithm has difficulty detecting small objects within an image, such as birds in a flock.

2.3 *DETR*

In DETR[11], the object detection problem is modeled as an image-to-set problem. Given an image, the model must predict an unordered set (or list) of all the objects present, each represented by its class, along with a tight bounding box surrounding each one. It is the first object detection framework to successfully integrate Transformers as a central building block in the detection pipeline[3]. DETR matches the performance of state-of-the-art methods, such as the well-established and highly optimized Faster R-CNN baseline on the challenging COCO object detection data set, while also greatly simplifying and streamlining the architecture.

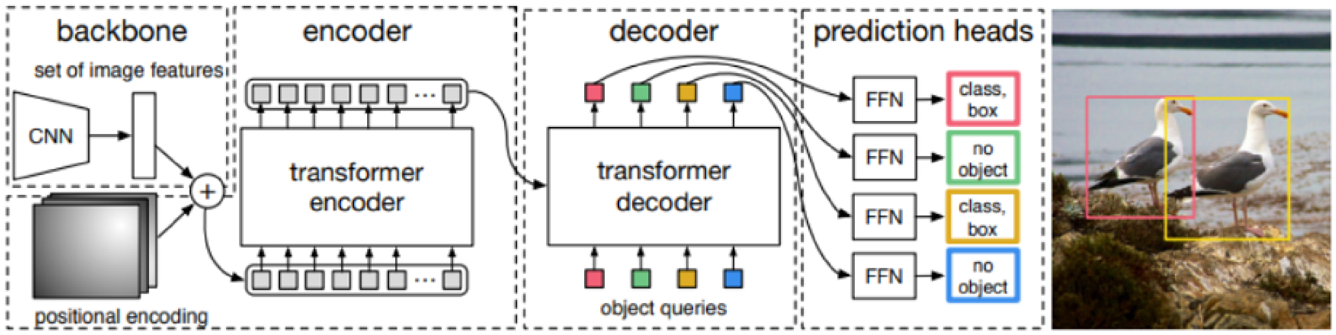

Figure 6: DETR Architecture: 1.a CNN backbone 2.an Encoder-Decoder transformer 3.a simple feed-forward network

The CNN backbone generates a feature map from the input image. Then the output of the CNN backbone is converted into a one-dimensional feature map that is passed to the Transformer encoder as input. The output of this encoder is the N number of fixed length embedding (vectors), where N is the number of objects in the image assumed by the model. The Transformer decoder decodes these embedding into bounding box coordinates with the help of self and encoder-decoder attention mechanisms. Finally, the feed-forward neural networks predict the normalized center coordinates, height, and width of the bounding boxes and the linear layer predicts the class label using a softmax function.

3 Data

The data was taken from the 2018 Data Science Bowl Featured Prediction Competition In Kaggle[1]: "Find the nuclei in divergent images to advance medical discovery". This dataset contains a large number of segmented nuclei images. The images were acquired under a variety of conditions and vary in cell type, magnification, and imaging modality (brightfield vs. fluorescence). The dataset is designed to challenge an algorithm's ability to generalize across these variations. Although It was a segmentation competition, some other authors published a bounding boxes file. The data contain two parts: a folder with all the images and a CSV with the bounding boxes in YOLO[10] format. I change the YOLO format to COCO during the preprocessing stage (DETR works only with COCO)[9].

4 Methods

We used transfer learning on the DETR model, fine-tuning the existing head[6]. DETR is based on the Transformer architecture. The Transformer architecture has "revolutionized" Natural Language Processing since its appearance in 2017. DETR offers several advantages over Fast-RCNN — simpler architecture, smaller networks (in terms of parameters), more throughput, quicker training. The Transformer architecture has pretty much replaced LSTM/RNN in sequence prediction tasks and is used heavily in Natural Language Processing (NLP). Figure 7 shows the DETR Transformer architecture with Encoders and Decoders.

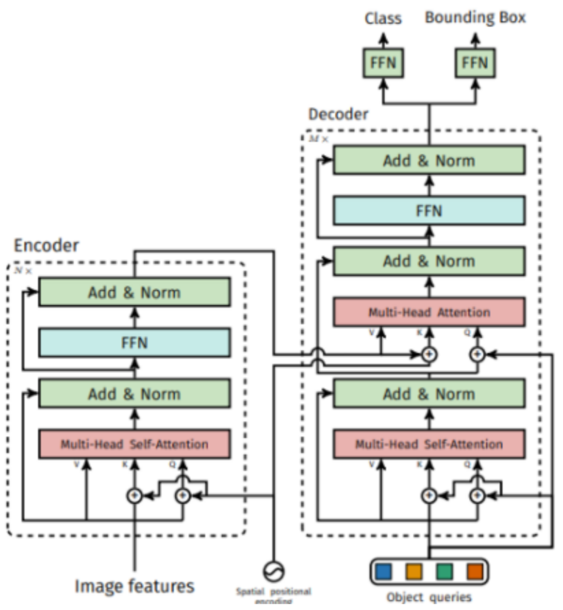

Figure 7: The bracketed notation indicates the shape of tensors. The output from the decoder is taken through a linear and an MLP layer respectively to generate the class and bounding box output.

An interesting object in DETR is the loss calculation, which is somewhat unique with the use of Hungarian Matcher and IoU[4]. A Hungarian matcher is used to find the most probable bounding boxes from the model concerning the target. Hungarian algorithm is well known combinatorial optimization technique. We fork the DETR Github repo[5] because of its special loss where it assigns one ground truth bbox to a predicted box using a matcher, thus when fine-tuning we need the matcher (Hungarian matcher as used in the paper) and also the function SetCriterion which gives Bipartite matching loss for backpropagation[3].

The initial DETR model is trained on the COCO dataset, which has 91 classes (+ 1 background class), we modified it to run on our 1+1 class. Image detection is not a classical task on biological data, different algorithms have pitfalls with how biological data structs. Models like YOLO aren't sensitive to relatively small objects, in addition, models like DETR have the lack of the query and “sequence memorization”. Comparing the more general class images of elephants or ducks, which are made up of different aspects like ears and legs, biological data, specifically, cells or nuclei, are all round and difficult to separate.

5 Experiments

We show that DETR achieves competitive results compared to Fast R-CNN and YOLO.

Our main model was to tune the DETR model that was trained on Resnet50 with 91 generic classes. We train DETR with AdamW setting the initial transformer’s learning rate to 104, the backbone’s to 105, and weight decay to 104 the backbone is with ImageNet pre-trained ResNet model from torchvision with frozen batch norm layers. We report results only with ResNet50 due to low computational power. To help learning global relationships through the self-attention of the encoder, we also apply random crop augmentations during training, we used Albumentations package[2] to do simple augmentation (rotation, cropping and color-changing). Network configuration :folds=5,classes=2,queries=50,null class coef=0.5, batch size=8,learning rate=2e-5,epochs=15(where a single epoch is a pass over all training images once). We also tried to increase the number of epochs but due to a time limit, we encourage a problem. Furthermore, we found the optimal solution by manually grid searching samples of different learning rates, batch sizes, and augmentations.

Bounding box loss; The second part of the matching cost and the Hungarian loss is Lbox that scores the bounding boxes. Unlike many detectors that do box predictions as a w.r.t. some initial guesses, we make box predictions directly. While such approach simplify the implementation it poses an issue with relative scaling of the loss. The most commonly-used L1 loss will have different scales for small and large boxes even if their relative errors are similar. To mitigate this issue we use a linear combination of the L1 loss and the generalized IoU loss. Overall, our box loss is:

$$Lbox(bi,\hat{b}(i))$$

$$\lambda iouLiou(bi,\hat{b}(i)) + \lambda L_1||b_i - \hat{b}_\sigma(i)||_1$$

The mean average precision (mAP) was used; mAP[8] is the mean average precision at different thresholds of the IoU between the ground truth and predicted segmentation and is calculated as

$$\frac{TP}{TP + FP + FN}$$

where t is the IoU threshold and TP, FP, and FN are true positives, false positives, and false negatives, respectively.The aggregated Jaccard index[14].

To compare the results, we run YOLO and Fast R-CNN using detectron2. We assign a train test splitting (80-20) to all models. The validation matrices were used to check the modal. All codes are in the attached Github repo.

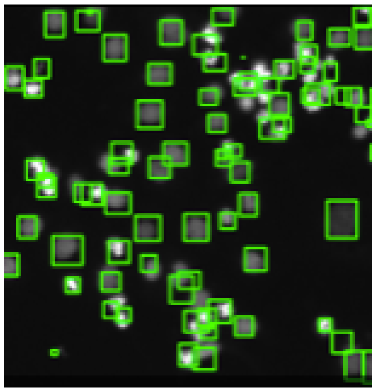
Figure 8: Example of YOLO results

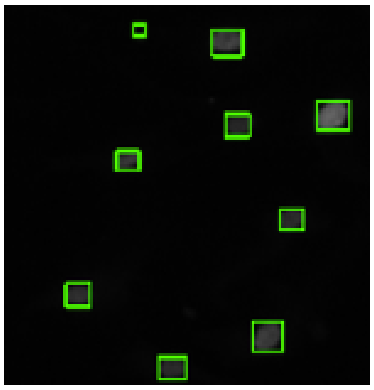
Figure 9: Example of Fast R-CNN results

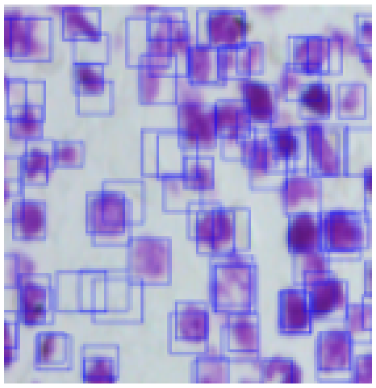
Figure 10: Example of Fast R-CNN results

6 Conclusion

Although we didn't get the outstanding result, we show that using a quick implementation of DETR we managed to get a good result with a small transfer learning that was based on a general everyday images dataset via resNet50. We can conclude that for getting fine but quick online results the best would be using DETR.

References

[1] 2018 data science bowl — kaggle. `https://www.kaggle.com/c/data-science-bowl-2018/data`. (Accessed on 03/03/2021).

[2] Albumentations: fast and flexible image augmentations. `https://albumentations.ai/`. (Accessed on 03/03/2021).

[3] detr_tutorial/dataset at master · thedeepreader/detr_tutorial. `https://github.com/thedeepreader/detr_tutorial/tree/master/dataset`. (Accessed on 03/03/2021).

[4] Easy object detection with facebook's detr — by mastafa foufa — towards data science. `https://towardsdatascience.com/easy-object-detection-with-facebooks-detr-d0bd9e4e53a4`. (Accessed on 03/03/2021).

[5] facebookresearch/detr: End-to-end object detection with transformers. `https://github.com/facebookresearch/detr`. (Accessed on 03/03/2021).

[6] lessw2020/training-detr: Unofficial colab on how to train detr, the intelligent object detector, with your own dataset. detr = detection transformer. `https://github.com/lessw2020/training-detr`. (Accessed on 03/03/2021).

[7] R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms — by rohith gandhi — towards data science. `https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e`. (Accessed on 03/03/2021).

[8] rafaelpadilla/object-detection-metrics: Most popular metrics used to evaluate object detection algorithms. `https://github.com/rafaelpadilla/Object-Detection-Metrics`. (Accessed on 03/03/2021).

[9] Taeyoung96/yolo-to-coco-format-converter: Yolo to coco annotation format converter. `https://github.com/Taeyoung96/Yolo-to-COCO-format-converter`. (Accessed on 03/03/2021).

[10] Yolo for detection of bounding boxes - tensorflow — kaggle. `https://www.kaggle.com/vijaybj/yolo-for-detection-of-bounding-boxes-tensorflow/data`. (Accessed on 03/03/2021).

[11] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. 2020.

[12] R. Girshick. Fast r-cnn. 2015.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[14] X. Mo, K. Tao, Q. Wang, and G. Wang. An efficient approach for polyps detection in endoscopic videos based on faster r-cnn, 2018.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. 2016.