KU LEUVEN

Exploiting Spatial Relations for Object Detection with Transformers

Oliver Schamp

Thesis submitted for the degree of Master of Science in Engineering: Computer Science, option Artificial Intelligence

Supervisor:
Prof. dr. ir. Tinne Tuytelaars

Assessors:
Prof. dr. ir. Marie-Francine Moens
Ir. Cedric Picron

Assistant-supervisor:
Dr. ir. Jens Bürger

Academic year 2022 – 2023

Preface

The writing of a thesis is rarely owed to one person alone. In this section, I would like to acknowledge all of those who have helped me complete this project, which I have thoroughly enjoyed developing. First and foremost, my gratitude goes out to my mentor Jens Bürger and promotor Tinne Tuytelaars for their excellent insights into the field of not just object detection, but of AI in general. Without their help and suggestions during our biweekly meetings I would have found this task much more difficult.

I would also like to say thank you to the team at Segments.ai, who provided me with a free license to their software.

This work builds upon previous the investigations into foosball object detection of Bert Vanelsande and Daan Seuntjens, whom I thank for their resources and inspiration. The training and evaluation of RetinaNet in particular came from a modified version of Bert's code.

Last but not least, I would like to thank my parents for supporting me throughout this master's year and always being prepared to help me if needed.

Oliver Schamp

Contents

Abstract

There are multiple ways to identify what an object is in a scene. The most obvious way would be to look at the appearance of the object and return an object class based on the visual appearance of the object. However, the position of the object in an environment can also help us to recognise objects when the superficial object features become difficult to distinguish. Doing this, we are able to identify objects in an image or in our environment not just by their appearance, but also their involvement with their surroundings. Despite this, many cutting-edge object detectors still only rely on feature representations [34].

This work investigates the performance of object detectors on the detection of objects in images of foosball tables. Identifying figures, balls, goals and tables is intended to be part of a larger project to develop an autonomous foosball player.

This work looks at not just identifying objects by their surroundings, but by specifically identifying objects based on spatial patterns and regularities. To do this, the transformer-based DETR model is used. The attention mechanisms present in DETR help to classify objects not only by their superficial features, but also by their participation in structural regularities. This is seen in DETR's performance on synthetic data during experimentation.

In this thesis, a train and validation set consist of 310 and 50 hand-labelled frames taken from foosball videos. A test set is made from 2000 synthetic images, that serve as abstract images of foosball tables. Sinusoidal functions are used as input to the synthetic data generation to produce a consecutive sequence.

DETR is trained and compared alongside RetinaNet, which is used as an example of an object detector which relies on feature representations alone. First, the models are trained on annotations where the classes involved are all visually different. Second, the models are once more trained on annotations where the only difference between some of the annotated classes is the position of their corresponding class objects in the scene, both in an absolute sense and in a relative (to other objects) sense.

The results show that DETR does exploit spatial regularities, as much higher performance is observed on the test set annotated with positional classes when compared to RetinaNet. Analysis of DETR's learned object queries and attention maps suggests that this increased performance is due to a mixture of two things. One, DETR recognises the specific areas that objects associated with each class appear in throughout the training data through trained decoder queries. Two, DETR recognises and exploits patterns and spatial regularities in the training data using the attention mechanism.

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

MLP	Multi-Layer Perceptron
FFN	Feed-Forward (Neural) Network
NN	Neural Network
CNN	Convolutional Neural Network
IoU	Intersection over Union
GIoU	Generalised Intersection over Union
AP	Average Precision
mAP	mean Average Precision
mAR	mean Average Recall
H	Image Height
W	Image Width
TP	(Number of) True Positives
FP	(Number of) False Positives
FN	(Number of) False Negatives
TN	(Number of) True Negatives
CV	Computer Vision

Important Semantic Note

In this document, 'Figure' with a captial F refers to the images of the report. An uncaptialised 'figure' always refers to the figures on the foosball table, which are meant to represent football players.

Chapter 1

Introduction

1.1 Overview

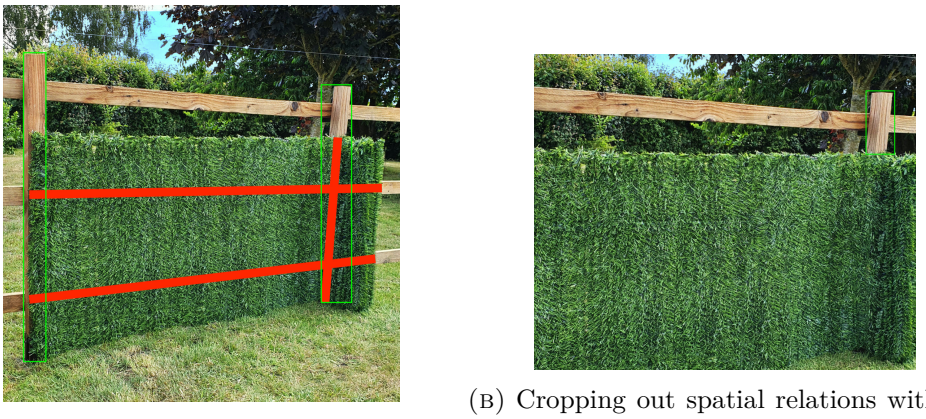
Over recent years, there has been an increasing interest in object detection for sports analysis. Controversy is ever-present in sport, and methods such as object detection and tracking can help justify contentious decisions. Object detection and tracking also helps machines to better understand the scene that they observe, which is crucial when creating an intelligent system able to make sophisticated decisions based on the situation. To develop effective physical systems dependent on computer vision for information retrieval, such as self-driving cars, one must extract as much relevant information as possible from the perceived situation, and then one must employ a reasoning system to return decisions based on the annotated data. This thesis focuses on the first problem of extracting data from a scene, assessing the task of of annotating video frames with bounding boxes.

The KU Leuven institute for AI would like to develop an autonomous foosball (table football) player, and naturally, to do so this player must be able to recognise its environment and make decisions accordingly. As such, the scenes/situations present in this thesis will be frames taken from foosball matches, aimed at object detection of the figures, football, goal and table, which are the objects that typically affect how a game proceeds.

From the perspective of a human, if an object in our field is occluded or blurry, we can often use the environmental context around it to still accurately recognise the object's location and type. For example, consider a task where the aim is to detect only the vertical beams of a wooden fence, most of which is occluded by a cover (Figure 1.1 - the green bounding boxes illustrate possible classifications by an observer). In Figure 1.1a, the vertical beam on the left extends all the way to the ground, and some horizontal beams are seen extending out from either side of the cover. This context allows for an extrapolation of the spatial structure of the fence behind the occlusion and a correct detection of the vertical fence beam on the right of Figure 1.1a. If Figure 1.1a is cropped to remove this spatial information (while keeping the entirety of the second beam's bounding box inside the frame, seen in Figure 1.1b) it becomes very difficult to detect the beam correctly, even as a human

observer.

This thesis aims to apply deep learning to image data to specifically explore whether an attention-based model can exploit spatial relations to detect objects by extrapolating upon learned patterns. The assumption of these patterns, as in Figure 1.1, should give benefits to performance when data is occluded or blurred out. The specific spatial relations from the context of a game of foosball are listed in Section 1.3.



(A) Spatial relations

(B) Cropping out spatial relations without cropping the object to be detected

FIGURE 1.1: Effect of spatial relations on the detection of vertical fence beams [31]

1.2 The game of foosball

Figure 1.2 shows the appearance of a typical foosball table. In this game, you have two adversarial teams, each with a different colour (in this case red versus black).

FIGURE 1.2: The layout of a foosball table [7]

Each side aims to get the ball in the opponents goal, by striking the ball while rotating their rows of football player figures. To do so, it is necessary to move the ball between your rows of figures, and past the opponents rows. Like most games, luck is

a factor, but typically higher success rates are found with some measure of skill. The skills required of a good foosball player are good hand-eye coordination, an extensive muscle memory of how to play the game, a fast reaction time, and an ability to make optimal decisions inside very small time periods. Computer vision systems would help an autonomous player improve the chance of making the optimal decision for every scene. An example of such decision making would be if an autonomous player is about to strike a ball with one of its figures, but a vision system detects an opposing figure in the way of its shot, the autonomous player can decide to bail on the original action and instead pass or hold the ball.

1.3 Problem statement

When observing Figure 1.2, one notices many spatial regularities (some similar to Figure 1.1). The relevant perceivable spatial regularities present in all foosball tables are as follows:

1. A fixed amount of bars with figures on (8 in total, 4 for each team)
2. A pattern in the amount of figures associated along each bar (1-2-3-5-5-3-2-1)
3. A pattern in the colour of the figures (each bar always has the same colour of figure, the figures usually alternate colours when moving vertically along Figure 1.2)
4. Even spaces between figures on each bar
5. Even spaces between each bar and its neighbouring bars
6. Two goals can be found in the middle of either end
7. All figures can be found inside the table

The perspective of the camera can modify some of these spatial relations. With changing perspective, the image data shows different patterns that are nevertheless still all representative of rules 1, 4, 5 and 6. For example, Figure 1.2 represents rule 5 as a decreasing width between bars as distance from the camera to the table increases. If the table had higher walls, it might obstruct the camera's line of sight when detecting some of the bars on the table. Therefore, this thesis only uses very similar perspectives, with more elaboration on this in Section 4.1.

The most prevalent spatial relations in the game of foosball are associated with the figures and the goal. When detecting the figures on the foosball table, some figures could be blurred, or occluded by other figures. The goals are always present in the same place at the same size, but can have different features and colours, making the information required to classify the goal object mostly based on spatial regularity. This thesis aims to analyse a deep learning model (trained on hand-labelled data) to find out the extent to which the model exploits the spatial relations listed above in the detection of the goal and figure objects.

Chapter 2

Background

This section is comprised of two subsections. The first subsection explains the general principles of machine learning that are important to understand for this thesis. The second subsection explains principles specific to object detection that are used in later sections.

2.1 Principles of machine learning

NOTE: See Appendix A for an introduction to neural networks.

2.1.1 Convolutional Neural Networks

A CNN possesses its trainable weights in the form of NxN filters, where $N << (W \; and \; H)$. These filters are convolved with the image to produce a new image representation, where the new representation dimension is determined by the size of the filter, the number of channels (number of filters present), the 'padding' used and the 'stride' used. CNNs can have multiple layers, as shown in Figure 2.1. The intermediate layers of the CNN are sometimes called 'feature representations', as trained filters will extract type of edges, curves or more complex features.

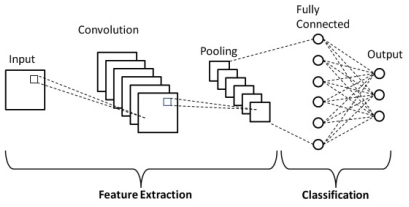


FIGURE 2.1: Simple CNN Architecture [23]

A standard CNN for image classification typically has pooling layers to condense feature representations, and an MLP at the end which takes a flattened feature representation as input. CNNs were first introduced by Yann LeCun in the Neocognitron and LeNet papers [19]. In the past decade, they have gained status as the most popular deep learning model architecture for computer vision. Although each

layer focuses on local relations, a deep CNN architecture and a large enough filter dimension ensures that each particular output that is flattened for the MLP head can be influenced by any pixel value present in the input image.

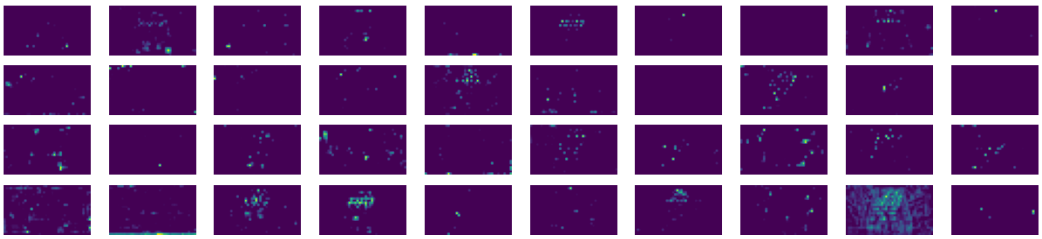


FIGURE 2.2: Some of the CNN features extracted from Figure 1.2

2.1.2 Transformers and attention

This thesis investigates the benefits of using a transformer architecture with a CNN backbone for an object detection task. In the field of CV, Transformers have seen success on tasks such as image classification, with models such as the Vision Transformer [5]. Transformers have seen great success in natural language processing because of the principle of 'attention', through which the model can learn to recognise underlying patterns present in its input data [32].

Figure 2.3 shows a high-level diagram of a standard transformer architecture. In this figure, there are two dotted rectangles encircling the 'encoder' and 'decoder'.

The attention mechanism coupled with a near-perfect memory help transformers to understand sentence structures and meaning very efficiently. The outstanding success of transformers has led to large language models such as ChatGPT, which is now widely used worldwide [21]. As it is easier to understand how the transformer works in the context of natural language processing, this section will focus on how transformers process and understand language. Parallels between language understanding and exploiting spatial relations are listed in Table 2.1 and will also be expanded upon in Section 3.3.

Let us assume a machine translation task, where one has a sequence of S words from which to produce a corresponding output sequence in another language. First, the words are transformed into word embedding vectors (size 1xE). These vectors are passed in parallel to the transformer encoder. A single transformer encoder block first contains a multi-headed attention layer, which takes the input vectors an generates a vector output for each word embedding in the input. This output is representative of the relation between that word and the words around it.

Attention works in transformers via the Queries (purple), Keys (orange) and Values (blue), displayed in Figure 2.4. Figure 2.4 shows a detailed diagram of the entire process occurring inside the 'Multi-head attention' blocks of Figure 2.3.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

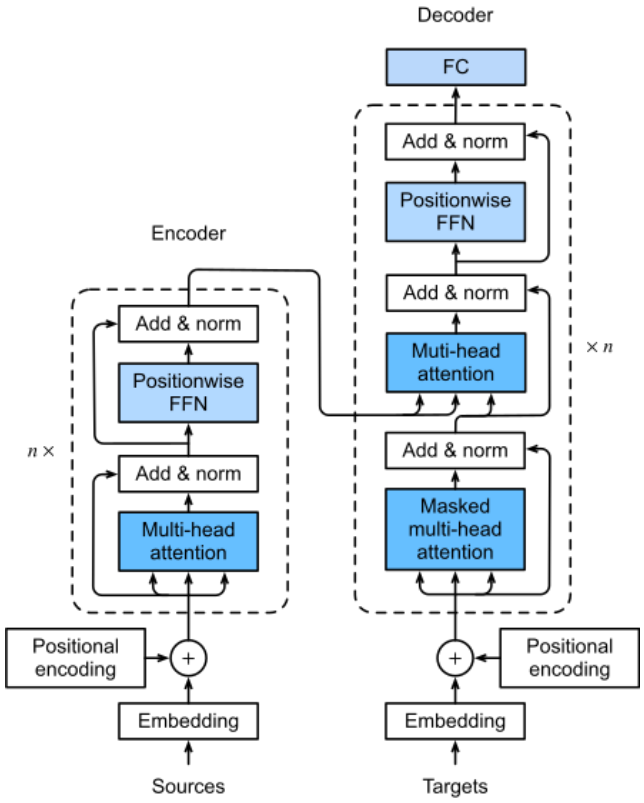


FIGURE 2.3: Transformer architecture [11]

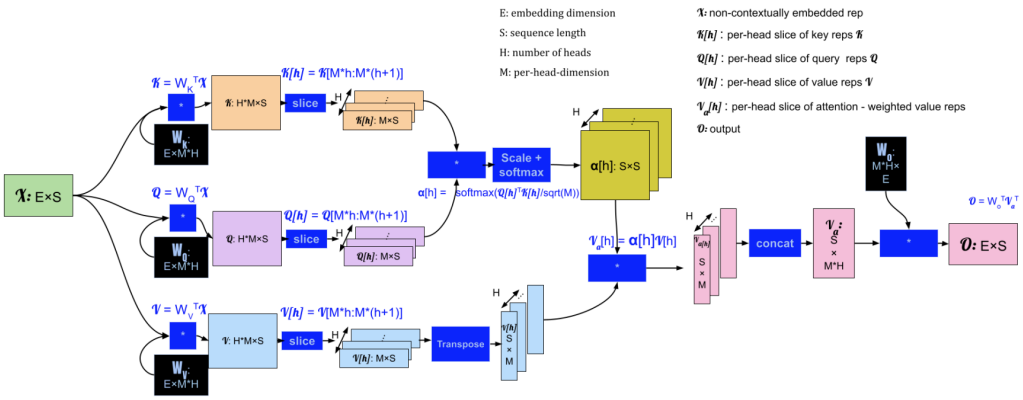


FIGURE 2.4: Multi headed attention [10]

As input, there come S words of E-length word embeddings, creating an ExS matrix. This matrix is copied three times, and each copy is multiplied by a set of trainable weights W_Q, W_K or W_V (of dimension MxE, where M is the dimension of the attention output) to produce either a Q, K or V matrix of dimension MxS. Each word in the sequence thus has its own Query, Key and Value vector. The dot product

of the Query and Key vectors is used to calculate a 'score' for each word, which aims to model a relation of one word to other words (for example, someone's name, and their pronoun might have high scores for each other). The role of the queries is to specify a 'direction', while the keys react to the queries. For every query, there are a corresponding S keys that calculate the scores for that query. To normalise these scores, a softmax function (Equation A.2) is applied to all scores calculated for each query. This results in an SxS matrix (Figure 2.4, yellow), which contains the scores of all S words in the sequence against each other.

This score matrix is the numerical representation of the attention that the transformer gives to different parts of its input. A high score between words is symbolic of a stronger association between them in the context of the sentence. Therefore, when this matrix is multiplied by the V matrix (which just like the Q and K matrices, is a weighted linear projection of the input vectors) it acts as a weighting on the word inputs. This process is written in Equation 2.1.

However, often sentences contain more than a single type of contextual relation. Therefore, the attention process is done in H parallel processes, called 'heads'. This now results in multiple outputs, one for each head (Figure 2.4, pink) which are then concatenated and projected onto the original dimension of the input via a linear layer of fully-connected neurons. Each vector output of the multi-head attention block is then passed through an MLP (all parallel MLPs contain the same weight values) in parallel. This completes the encoder block, and the output of the MLP is the also the output of the encoder block.

The decoder works much the same way as the encoder, and it is at the output of the decoder that the final translated sentence is produced. To generate the output sentence, the decoder receives a single input vector, in an NLP case the vectorised <START> token. This target vector is first passed through a self attention layer, then a cross-attention layer. In the cross-attention layer, rather than training weights to produce Q, K and V matrices, only trains the weights to produce a Query matrix. For the K and V matrices, the encoder output is used instead. This introduces the input sequence into the decoder. The output of the cross-attention is passed through an MLP, before returning as output a vector of floats.

In NLP, this vector of floats is transformed into an output word via a linear layer followed by a softmax layer. This output word is then fed back as input to the decoder in an autoregressive fashion until the transformer outputs and <End of Sentence> token. In this thesis, the decoder used for a computer vision application has some differences to the NLP decoder, most notably that the decoder used in DETR [4] is not autoregressive, but decodes all targets in parallel (Section 3.3).

The attention mechanism is what makes a transformer adapted for a computer vision task theoretically able to recognise spatial relations inside an image. This thesis aims to analyse what associations the transformer architecture draws from image data by replacing the machine translation aspects introduced previously with those more suited to object detection (Table 2.1). This will further be elaborated on in the literature review of DETR in Section 3.3.

Type	Machine Translation	Object Detection
Inputs	Word embeddings	Image features
Outputs	Word embeddings	BBox coords
Decoding	Autoregressive decoder	Parallel decoding
Decoder queries	Generated queries	Learned queries
Loss function	Categorical cross-entropy loss	Hungarian matching loss

TABLE 2.1: Adapting a machine translation task into an object detection task

2.1.3 The Hungarian Algorithm

The Hungarian algorithm, also called the Munkres algorithm, is a method of selecting the optimal assignments of a number of N labels to a number of N variables. In the case that no label can be given to more than one variable, the Hungarian algorithm attempts to minimise the sum of all individual costs of assigning each label to a variable. To do this, the Hungarian algorithm constructs an NxN matrix of costs pertaining to each individual variable-label combination, and follows the following steps:

1. Row reduction: subtract the row minimum from each row
2. Column reduction: subtract the column minimum for each column
3. Check for an optimal assignment on whether you can draw N lines through all zeros in the NxN matrix
4. If optimal assignment, skip steps 4, 5 and 6. If no optimal assignment, find the smallest value uncovered by a line and subtract it from all uncovered values.
5. Add this value to all values at the intersection of two lines
6. Test again for an optimal assignment, if no optimal assignment, repeat steps 4 and 5.
7. Choose an assignment of zero-values in the matrix such that each row and column of the matrix contains only one chosen zero.
8. The locations of these zeros are the locations of your optimal assignments in the original matrix.

One use of the Hungarian algorithm is for solving bipartite matching problems in object detectors, as it helps find the optimal choice of outputs to display from an object detector relative to the ground truths of a scene.

2.1.4 Accuracy, Precision and Recall

Accuracy, precision and recall are all metrics used to evaluate the performance of not only deep learning models, but any form of classifier.

Accuracy

Accuracy is defined as the fraction of predictions that a model returns that are correct according to the ground truths.

$$Accuracy = \frac{Correct\ Classifications}{Total\ Classifications\ Possible} \tag{2.2}$$

Precision

The number of 'correct predictions' can be thought of as the number of 'true positives', where the ground truth specifies an output, and the model returns a positive for the desired output. Therefore, 'false positives', 'true negatives' and 'false negatives' can also exist. Precision is calculated in Equation 2.3.

$$precision = \frac{TP}{TP + FP} \tag{2.3}$$

Precision is useful for when minimizing false positives is the goal, because a precision value is effectively a ratio between the true positives and false positives returned. A precision value of 1 is returned when there are no false positives, i.e., when the model never predicts something to be present when it is not.

Recall

Recall is also often used to evaluate model outputs. For recall, the importance is placed on minimizing false negative results. A recall of 1 is achieved when a model returns no false negatives, i.e. the model always detects all classes present and therefore never ignores a classifiable object when it is present.

$$recall = \frac{TP}{TP + FN} \tag{2.4}$$

2.2 Object detection concepts

2.2.1 Bounding boxes and object detection

Object detection means recognising and locating a specific object inside a scene. In this thesis, the *scene* will be an image of a foosball table during play. *Bounding box detection* is where the model draws rectangles around the objects it detects to highlight their location inside the scene, whereas *panoptic segmentation* looks to classify each pixel in the image as associated with either an object or the background. This thesis explores object detection by bounding box prediction.

2.2.2 Generalised IoU

When drawing bounding boxes, one must have a method to compare the ground truth bounding box 'A' to the predicted bounding box 'B'. The Generalised Intersec-

tion over Union (GIoU) can be used to measure how well a drawn box overlaps with and matches the ground truth.

(A) IoU (B) GIoU

FIGURE 2.5: IoU and Generalised IoU

The calculation for just the IoU is:

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN} \tag{2.5}$$

The reason the IoU is so prevalent in the evaluation of bounding boxes rather than another calculation (which could be, for example, the fraction of the ground truth bounding box also found inside the predicted bounding box) is that even if the predicted bounding box fully envelopes the ground truth, the IoU still punishes bounding boxes that are drawn excessively large over their ground truths. If one thinks of TP, FP and FN in terms of area, the IoU can be thought of as a combination of precision and recall for bounding boxes (Equation 2.5). Another advantage of the IoU is that it is scale invariant.

The IoU can still be improved to become a Generalised IoU (GIoU). Whereas a simple IoU would give a value of 0 for all non-overlapping boxes regardless of how close they find themselves to each other, using the Generalised IoU gives negative scores proportional to the distance of the prediction from the ground truth. Therefore, the GIoU provides a correlated relationship between the L2 loss of the bounding box corners and the bounding box loss [25].

The GIoU draws a box around the bounding boxes present (Figure 2.5b), and uses the area C of the total rectangular space to calculate a metric that returns large negative scores for boxes that are long distances apart (Equation 2.6).

$$GIoU = IoU - \frac{|\frac{C}{|A \cup B|}|}{|C|} \tag{2.6}$$

2.2.3 mean-Average Precision and mean-Average Recall

An important concept to mention before introducing average precision is that of IoU thresholds. IoU thresholds allow for the conversion of IoU scores into precision and recall values. If the IoU score returned is above a certain threshold, then the object detection is returned as a True Positive provided that both boxes have matching classes. For example, if an IoU threshold value is 0.7, but an object detector

draws a box that only scores 0.69 against the ground truth, that prediction will be logged as a false negative regardless of if the class labels match or not.

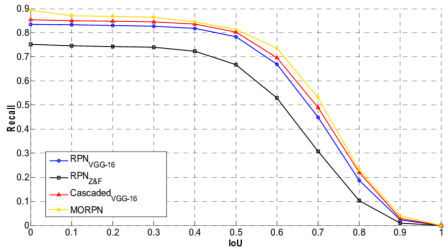


FIGURE 2.6: Effect of IoU threshold on Average Recall [33]

As one can imagine, changing the IoU threshold has a certain effect on the average precision an recall values. A low IoU threshold results in fewer False Negatives, but also more False Positives. A high IoU threshold results in many False Negatives, but fewer False Positives. This gives a precision-recall curve as one increases the values of an IoU threshold, as recall generally decreases and precision generally increases with increasing IoU threshold. Figure 2.6 shows how recall is affected by increasing IoU threshold.

The mean-average precision (mAP) and mean-average recall (mAR) is the mean of multiple Average Precision/Average Recall values returned for several unique IoU thresholds.

2.2.4 COCO evaluation format

This thesis uses the evaluation metrics of the COCO object detection dataset, which are 12 variants of Average Precision and Recall. They are listed in Table 2.2.

metric	IoU	area	maximum detections
mAP	0.5:0.95	all	100
AP	0.5	all	100
AP	0.75	all	100
mAP	0.5:0.95	small	100
mAP	0.5:0.95	medium	100
mAP	0.5:0.95	large	100
mAR	0.5:0.95	all	1
mAR	0.5:0.95	all	10
mAR	0.5:0.95	all	100
mAR	0.5:0.95	small	100
mAR	0.5:0.95	medium	100
mAR	0.5:0.95	large	100

TABLE 2.2: COCO evaluation metrics

An IoU '0.5:0.95' means that this value is a mean-AP/AR, calculated as the mean of AP/AR values coming from IoU thresholds inside the range of 0.5 to 0.95. 'small',

'medium' and 'large' refer to categories that describe the areas of the bounding boxes. The 'maximum detections' describes the maximum number of detections considered for each class before calculation of the mAP/mAR metric.

Chapter 3

Literature Review

3.1 Previous works on player detection in sports

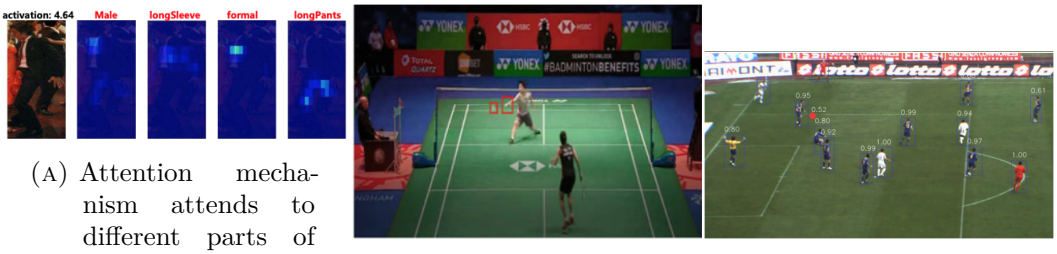
There have been many previous works centred around computer vision and deep learning for image classification and object detection in sports. One of these studies [2] used the SF-YoloV5 object detection algorithm to track the rackets and shuttlecocks of a badminton match in order to predict shots before execution, and classify them after execution. This work differs to those compared in this thesis as the SF-YoloV5 algorithm was only used to generate region proposals as part of a two-stage detector, whereas this thesis focuses only on one-stage detectors. Two stage detectors typically first return bounding box 'region proposals' before classifying these regions in the second stage of detection.

Another research paper found good results using the YoloV2 algorithm for the tracking of handball players [3]. This paper also investigated the performance of three progressively more advanced tracking algorithms, all based on using the Hungarian assignment algorithm to batch detected bounding boxes to their most likely recorded 'track'. A study on the tracking of football players [16] used the YoloV3 architecture without fine-tuning (the players were detected by the original COCO dataset class 'person') to detect players before using the SORT algorithm (a combination of the Kalman filter and Hungarian algorithm) to track the paths of these detected players.

For combined player and ball detection, J. Komorowski et al [18] use a fully convolutional arcitecture rather than a Yolo architecture based design. Their model used Feature Pyramid Networks, and returned a confidence map along with bounding box coordinates for the players and ball in the scene. The paper uses a CNN to extract features from the scene, and the model ends with three separate outputs: a ball confidence map, a football player confidence map and a bounding box regression map, with four channels to represent the bounding box coordinates. Thresholding is then used to choose which of the bounding box proposals best fit the confidence maps of the players and the ball. The good performance returned by the paper and the close application link to foosball, this paper inspired an FPN-based object detection as the point of contrast for behavioural and performance analysis against the Transformer architecture.

There has also been research into player and ball detection in football using CNNs as part of player and ball tracking, which have returned good performance in classifying which action sequences found in consecutive frames belong to moving players or balls. [17] used a fine-tuned VGG architecture to evaluate the blobs returned by separate ball detection and player detection algorithms. [35] compared a one-stage (YoloV3) and a two-stage (Faster-RCNN) object detector with the Kalman filter. While this thesis does also evaluate the detection of the ball object in the foosball table, it may be that a separate algorithm for ball detection proves more reliable than a transformer model due to the varied spatial contexts and loose spatial constraints the ball finds itself in, in comparison to other detectable foosball objects.

Using the attention mechanism has also been explored for image classification. Using a CNN classifier combined with an attention head, [36] classifies images with great performance. Upon visualisations of classified images, the attention modifies itself according to the object. For example, when classifying an object as a person, the model attends mostly to the face of that person.



(A) Attention mechanism attends to different parts of the image based on class [36]

(B) Badminton object detection [2]

(C) FootAndBall object detection [18]

FIGURE 3.1: Important figures from different papers

3.2 RetinaNet

As this thesis analyses how transformers exploit spatial relations, it is useful to have another model to compare and contrast behaviours with, to emphasise the object detection behaviours that come with exploiting spatial relations. RetinaNet [20] is a one-stage object detector consisting of Feature Pyramid Networks combined with a CNN-style architecture, giving similar performance on the COCO dataset to DETR. In the DETR paper, the authors claim to receive an AP_{50} of 62.4 on the COCO validation set, and for RetinaNet the authors claim to receive an AP_{50} of 61.1 on the COCO test-dev set.

The important features of RetinaNet are twofold: their 'focal loss' and the use of Feature Pyramid Networks. A Feature Pyramid Network is a method of combining higher and lower level features from the backbone. The FPN does this using skip connections between the lower and higher layers of the backbone and and upsampled pyramid. This produces feature maps of constant size that represent both represent both fine-grained specific features inside a scene, and also wider-reaching, larger

abstractions. For RetinaNet, this FPN representation is then passed to a CNN, which is split into two convolutional outputs which are eventually flattened and passed to two MLPs. These MLPs output the bounding box coordinates and their respective classes.

3.3 DETR

DETR was first introduced in 2018 by Facebook Research. As seen in Figure 3.1a, attention can be used to observe different spatial features depending on the classification objective. For example, if the model would like to classify “long pants", the attention map returns larger magnitudes around the legs of the person. If the model would like to classify “male", or more specifically a class such as “sunglasses", the attention focuses more on the defining facial features of that person.

In foosball, one must consider the point at which a metal bar attached with multiple wooden pegs becomes a row of foosball figures, part of a larger ‘team’ controlled by a player playing the game of foosball. The criteria to move from a metal bar, and from wooden pegs attached to such a metal bar, into foosball figures, is at the abstract level entirely positional, related to those specifications listed in the introduction. As explained in Section 2.1.2, the attention mechanism is key to transformers and is the reason for their recent successes in the field of NLP. Because transformers can recognise relations between words via multi-headed attention, they should also recognise relations between image features, which could then recognise spatial consistencies if beneficial to the object detection task. Therefore, using a transformer architecture like DETR could help increase the performance of foosball object detection compared to CNN-based models by implicitly recognising and looking for those spatial relations present whenever the model must identify classes such as the figures and the goal.

The architecture for DETR is shown in Figure 3.3. Figure 3.3 is very similar to Figure 2.3, but the difference arrives in the context of the inputs and outputs (briefly explored in Table 2.1). The main architecture is relatively simple, with the encoder and decoder blocks being identical in structure as with the original transformer. When using DETR, an image is given as input to a transformer architecture with two MLP heads, one providing a classification and one providing a bounding box coordinate specification. Figure 3.2 shows the intricacies of how the image data passes through a CNN backbone, before coming as input to the transformer encoder, which contributes cross-attention to output a list of classifications and bounding boxes dependent on trainable decoder queries.

3.3.1 ResNet Backbone

Because of the quadratic cost of the attention mechanism, DETR starts with a CNN backbone to extract image features. For this thesis, DETR is used with a ResNet50 [13] backbone. First, the inputs to the ResNet backbone are padded to the maximum height, H, and width W found in the input data. The backbone outputs 2048 image features of $\frac{H}{32}$x$\frac{W}{32}$. Some of these features for a foosball table are

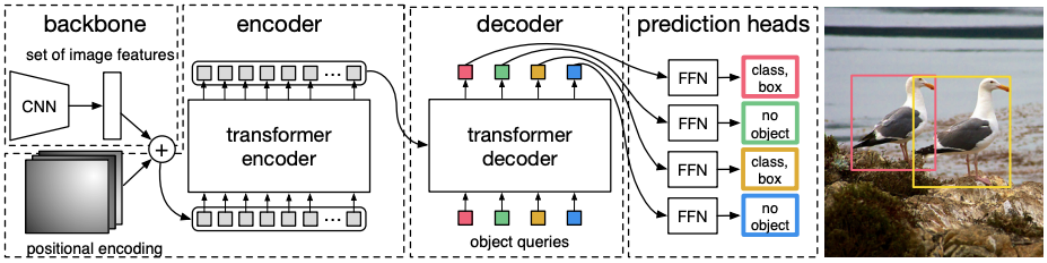


FIGURE 3.2: How information flows through the DETR model [4]

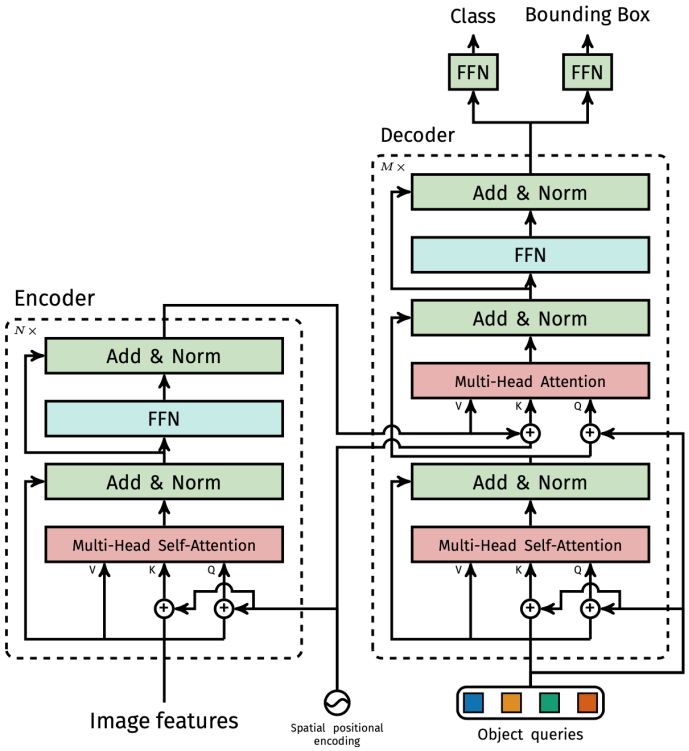


FIGURE 3.3: High-level diagram of the transformer architecture for object detection [4]

displayed in Figure 2.2, where it is noticeable that they are of much lower resolution than the original image. For future notation, $\frac{H}{32} = H_0$ and $\frac{W}{32} = W_0$. The number of output channels is then reduced to 256 by a 1D convolution. The image features are then flattened, and given to the transformer as H_0xW_0 input tokens of embedding dimension 256.

3.3.2 Positional Encoding

The positional encoding is the same sinusoidal function seen in the "Attention is all you need" [32] paper, generalised to images. In algorithm 1, the positional encod-

ings returned are vectors of length d_{model}, which is 256. The *pos* is the cumulative sum of the mask of the image feature in either the x or the y direction. The difference between this positional encoding and that from NLP is that while sentences are 1-dimensional, an image is 2-dimensional. Therefore, the final positional encoding is split into the concatenation of not just sine and cosine, but also the cumulative sums of the image feature masks in both dimensions pos_x and pos_y (which are both H_0xW_0 matrices).

Input : Sequence length $L = H_0 * W_0$, embedding dimension $d_{\text{model}} = 256$
Output : Positional encoding matrix P of shape $L \times d_{\text{model}}$

for $pos_x = 0$ **to** $L - 1$ **do**
  for $i = 0$ **to** $\lfloor d_{model}/4 \rfloor$ **do**
    $P[pos_x, 4i + 1] = \sin\left(\frac{pos_x}{10000^{2i/d_{\text{model}}}}\right)$;
    $P[pos_x, 4i + 3] = \cos\left(\frac{pos_x}{10000^{2i/d_{\text{model}}}}\right)$;
  end
end
for $pos_y = 0$ **to** $L - 1$ **do**
  for $i = 0$ **to** $\lfloor d_{model}/4 \rfloor$ **do**
    $P[pos_y, 4i] = \sin\left(\frac{pos_y}{10000^{2i/d_{\text{model}}}}\right)$;
    $P[pos_y, 4i + 2] = \cos\left(\frac{pos_y}{10000^{2i/d_{\text{model}}}}\right)$;
  end
end

Algorithm 1: Positional Encoding (image data)

This sinusoidal positional encoding generates $L = H_0 * W_0$ tokens of $d_{model} = 256$ length. In Figure 2.2, the maximum resolution of the image is 750*1333, giving L a value of 1008 after both dimension sizes are divided by 32, giving image features of resolution 24x42.

3.3.3 Parallel decoding an decoder queries

Unlike in the NLP application of transformers, the decoding in DETR is done in parallel. Each of the N queries is a trained vector of length $d_{model} = 256$, and they pass through the decoder at the same time, with each one therefore uniquely producing a class output and a bounding box output, while being able to attend to each other. Through multiple self-attention and encoder-decoder attention layers, these queries are trained. The outputs they produce are given as N vectors to the two output heads shown in Figure 3.3, which produce a classification and a set of bounding box coordinates respectively. Therefore, the maximum number of (class, bbox) outputs that DETR can produce per image is limited to a maximum of N.

3.3.4 Loss function

The authors of DETR describe their loss function as a 'direct-set bipartite matching loss', where 'direct set' refers to the fixed set of N classifications and bounding box coordinates returned, and 'bipartite matching' refers to the procedure of matching each N outputs to a ground truth in an optimal fashion. The loss function is a combination of the class error, the GIoU (Section 2.2.2), and the L1 loss between the box centres. To put it briefly, Equation 3.2 evaluates how well the bounding boxes and classes of the model output and the ground truth (where each (class, bbox) ground truth has been explicitly paired with only one (class, bbox) output of the model) agree with each other.

Usually, object detectors require multiple facets to their loss functions such as NMS to remove duplicate detections. However, DETR does not require this and instead designs its loss function to mitigate these issues. How the bipartite matching does this is by limiting each (class, bbox) output pair of the object detector to be matched with a single ground truth.

$$\hat{\sigma} = argmin_{\sigma \epsilon \Gamma_N} \sum_{i}^{N} L_{Hungarian}(y_i, \hat{y}_{\sigma(i)}) \tag{3.1}$$

$\hat{\sigma}$ is the optimal one-to-one matching arrangement that gives the lowest loss value, where σ represents a permutation of decoder outputs. The Hungarian algorithm (Section 2.1.3) is used to find this optimal bipartite matching.

$$L_{Hungarian}(y, \hat{y}) = \sum_{i=1}^{N} [-logp_{\hat{\sigma}(i)}(c_i) + 1_{c_i \neq \emptyset} L_{box}(b_i, b_{\hat{\sigma}(i)})] \tag{3.2}$$

The bounding box loss specifically is a combination of the GIoU and the L1 loss:

$$L_{box}(b_i, b_{\sigma(i)}) = \lambda_{iou} L_{giou}(b_i, b_{\sigma(i)}) + \lambda_{L1} ||b_i - b_{\sigma(i)}||_1 \tag{3.3}$$

3.3.5 Stacked encoder/decoders and auxiliary losses

The encoders and decoders of a transformer can be stacked as in the structure shown in Figure 3.4, which is also the encoder-decoder structure used for the implementation of DETR in this thesis.

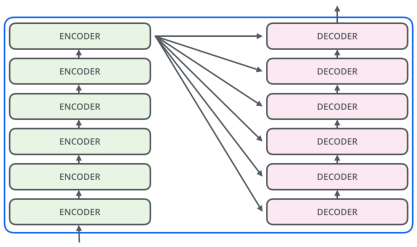


FIGURE 3.4: DETR uses 6 stacked encoder-decoders [1]

The output from each decoder is used during training to return a loss as if the decoder was the final decoder. These auxiliary losses help speed up the convergence of DETR during training.

Chapter 4

Experiments

This section details the experimental methods and results of training and executing DETR on frames gathered from YouTube, and synthetic frames showing an abstract representation of a foosball table.

DETR hypothetically has three separate criteria it can use to evaluate an object in a scene:

- **The superficial appearance of the object.** If an object possesses X features, it belongs to class Y. Example: if an object has a rounded head & green socks, it is a person.
- **The absolute position of the object in the scene.** If an object is located inside the X coordinate range, it belongs to class Y. Example: if an object is found in the top left corner of the scene, it is a person.
- **The relative position of the object in relation to other features/objects in the scene.** If an object is located in a certain location relative to X other objects and features, it belongs to class Y. Example: if an object is found standing next to a row of similar objects, it is a sociable person.

These experiments evaluate how much emphasis DETR places on each of these three possible criteria. The third characteristic in particular is the definition of exploiting spatial relations, and this section hopes to find evidence of the model taking advantage of regularities in relative space.

4.1 YouTube frames: train/validation sets

The data for the train and validation sets when training DETR and RetinaNet was collected entirely from YouTube videos [9] [8] [30] [7] [6] [29] [27] [15] [14] [28]. In these 10 videos, 6 unique styles of foosball table were identified (Figure 4.1). The motivation behind using a variety tables was that using different styles would encourage DETR to exploit spatial relations rather than superficial appearance features.

(A) (B) (C) (D) (E) (F)

FIGURE 4.1: The 6 different styles of table used in the hand-labelled data in the train and validation sets

There are two main layouts of foosball table in production, the American layout and the international layout. Those shown in Figure 4.1 are of the international layout, which shall be kept a constant variable throughout my experiments. This means that the number of figures on each bar (from top to bottom) stays as a consistent 1-2-3-5-5-3-2-1 formation.

In Section 1.3, the spatial relations most relevant to the figures were:

- Even spaces between figures on each bar, and the immutable number of figures present on every bar
- Even spaces between each bar and its neighbouring bars, and the consistency in there being 8 bars

To keep spatial regularities constant, a similar viewing angle and perspective were chosen for all the frames extracted. As seen in Figure 4.1, only views looking down on the table at an angle of elevation of roughly 45 degrees were chosen. In these perspectives, all bars of the table are clearly visible (except for a goalkeeper in Figure 4.1d).

The tables were initially labelled with bounding boxes each belonging to one of four classes: figure, ball, goal and table. Although the spatial relations involved do not include the ball, the ball was still added as a class to evaluate the general performance of each model.

I hand-labelled 360 unique frames, stratifying the data so that it contained 60 images for each style of table present in Figure 4.1. Using a random split, this data was separated into 310 train images and 50 validation images. The images were taken as frames video at a rate of 1 frame per second, which for a fast-paced game such as foosball often meant they often presented themselves as non-consecutive to the observer.

The software used to label this data was Segments.ai [26], an external tool for drawing bounding box annotations (among other functionalities). Using their interface, 360 images were labelled efficiently and imported from the Segments.ai website using Python script. The format for the image annotations differed between what Segments.ai gave me and what DETR accepted (DETR takes the COCO bounding box label format, Segments.ai gave me the labels in the YOLO-Darknet format), so after importing the data from Segments.ai, the data annotation files were converted into COCO bounding box labels before training began.

The annotating of the data was designed to preserve as many spatial relations as possible. Therefore, occlusions were handled by drawing the bounding box around

where the occluded object would have been. This sometimes led to many overlaps in bounding boxes, but allowed for a constant figure height and width on every bar regardless of surrounding environment. The same occlusion behaviour was extended to the ball class. The bounding box for the goal area was drawn with minimal overlap to its surrounding frame, to encourage the model to predict the location of the goal based on its relative position in the table. An example of the hand-labelling is shown in Figure 4.2.

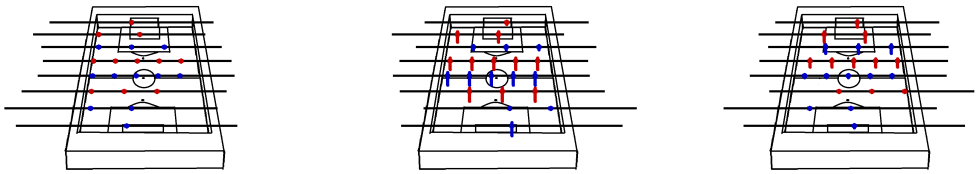
FIGURE 4.2: Example of a hand-labelled foosball scene [ball=yellow, figure=green, goal=blue, table=purple]

4.2 Synthetic data generation: test set

The problem with data collected from videos on the internet is that the angles of rotation of the figures around each bar, and the degree of horizontal shift on each bar become very difficult to measure. This means that the different individual states of a foosball table are not quantifiable. Furthermore, with video frames it is difficult to fully control the viewing perspective of the 'camera' which (as stated previously) should be a controlled variable. Therefore, synthetic data was generated to help quantify the varying states of a foosball table, and to test how capable DETR and RetinaNet were of abstraction.

To generate this data, a fixed background image of a basic foosball table outline was drawn in Microsoft Paint. Each bar with figures on was constructed as a separate image of overlapping rectangles and circles. Each bar image was then superimposed upon the background image, starting with the bar placed highest up on the image, to the bar placed lowest in the image. Before superposition, these bar images were also horizontally translated by a certain magnitude to create an illusion that the bar had been shifted by a player controller.

The lengths of the figures were determined from a variable of magnitude between 0 and 1, multiplied by a constant 'figure length' multiplier. Another constant 'figure height' multiplier (affected by the 'figure length' multiplier) gave the synthetic figures longer 'legs' than 'heads'. Figure 4.3 shows some examples of this synthetic data.



(A) (B) (C)

FIGURE 4.3: Three examples of different images in the synthetic data set

Each of the settings in Figure 4.3 is created by a different combination of 16 variables (8 for shifting the bars, 8 for 'rotating' the figures) between 0 and 1. A combination of 8 different ($y = sin^2(nx), n \in [2, 4, ...16], x = 0 to \frac{\pi}{2}$) sinusoidal signals were used as input for the horizontal shifts of the bars. The inputs to the sinusoids (x) were vectors of length 2000, giving 2000 discrete input combinations for the shifts of the horizontal bars. The same was repeated for the signals controlling figure rotations, or 'lengths'. Both the bar shifts and the figure lengths were then used to generate 2000 consecutive synthetic foosball images. Figure 4.4 shows how these signals created both the illusion of a consecutive sequence of synthetic frames, and also provided a large variety of unique table configurations.

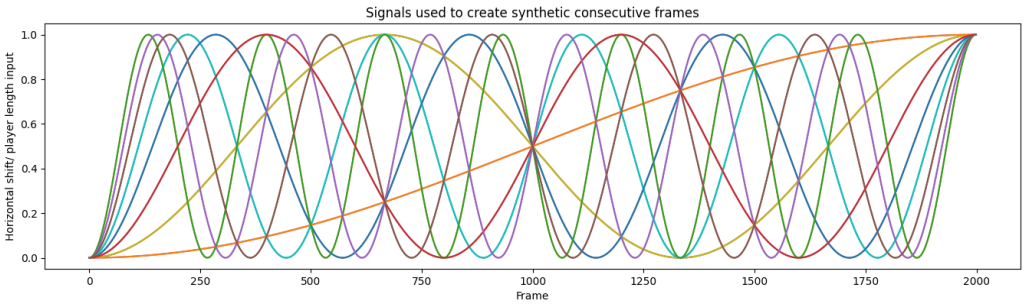


FIGURE 4.4: Sine inputs for generating 2000 consecutive frames of synthetic data

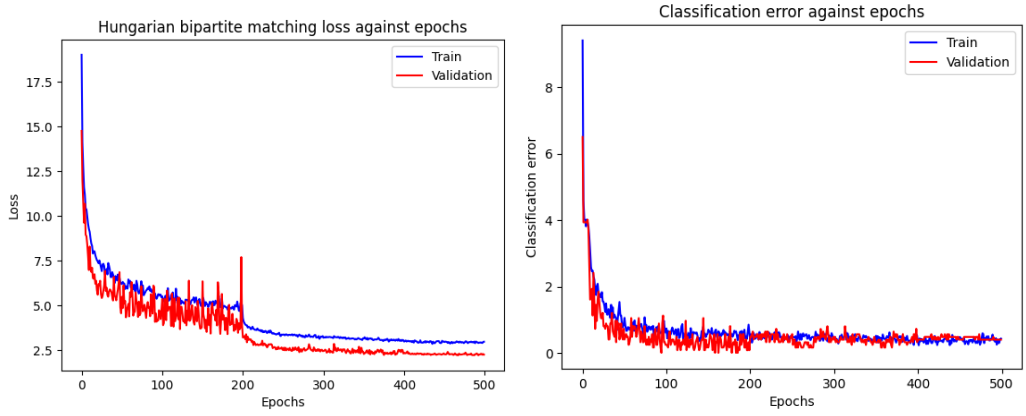
4.3 Fine-tuning DETR

The DETR training started by loading in the weights from the model zoo on DETR's GitHub page. This model of DETR was fine-tuned to output only four classes : the ball, the figure, the goal and the table. To fine-tune DETR, the MLP head for the classification was replaced with a new MLP head, containing an output layer size equal to the number of classes involved. The queries were also replaced with 29 randomly initialised queries (Xavier). The reason for 29 queries was to include 24 queries for all the figures, table and goal, plus 5 queries for the ball, and to give some flexibility to how the model can learn to use queries. Data augmentation is used as part of the DETR dataloader. These augmentations include randomly resizing the image, randomly cropping the image, and randomly flipping the image horizontally. All weights in the model were trained, none were frozen.

4.4 Evaluation of training

DETR was trained for 500 epochs with a learning rate of 0.0001, with the learning rate decreasing by an order of magnitude of 0.1 per 200 epochs. Figure 4.5a shows that as the loss on the training set decreases, so does the loss on the validation set. As the validation set loss almost always stays lower than the loss on the training set, one can hypothesise that the validation set has images which contain scenes where on average the objects are easier to detect.

Figure 4.6a shows the validation mAPs of DETR and RetinaNet after every epoch (see Section 2.2.3 for an explanation of mAPs/mARs). Conveniently, the figures and ball fall into the 'small' object category, while the goal falls into the 'medium' category and the table falls into the 'large' category. The model learns very quickly on how to detect the table, but struggles more when having to detect the smaller objects in the scene.

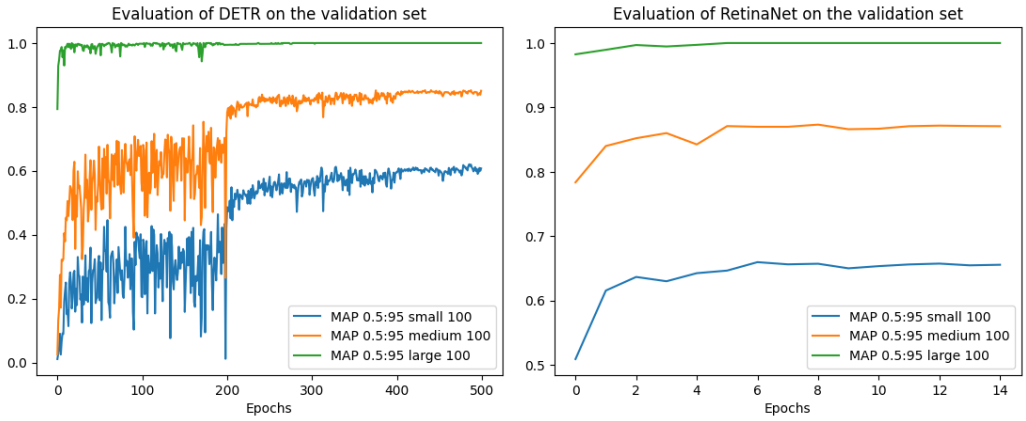


(A) Hungarian loss (Eq 3.2) (B) Class Error: $-log(P(class))$

FIGURE 4.5: DETR training curves for the train and validation sets in Section 4.1

4.5 Benchmarking against RetinaNet

RetinaNet was trained using the same COCO dataloader present in the DETR codebase. This was done to keep the data augmentations consistent between models. RetinaNet is much faster to converge than DETR, reaching a plateau in performance in only 15 epochs (Figure 4.6).



(A) mAPs in training, DETR (B) mAPs in training, RetinaNet

FIGURE 4.6: Validation set APs for differently sized objects throughout training. RetinaNet converges much faster than DETR

Threshold = 0.5	**DETR**	**RetinaNet**
AP (IoU=0.50:0.95)	0.763	**0.783**
AP (IoU=0.50)	**0.933**	0.900
AP (IoU=0.75)	0.820	**0.868**
AP (IoU=0.50:0.95) small	0.556	**0.593**
AP (IoU=0.50:0.95) medium	0.818	**0.869**
AP (IoU=0.50:0.95) large	1.000	1.000
AR (IoU=0.50:0.95)	0.798	**0.799**
AR (IoU=0.50:0.95) small	0.613	**0.621**
AR (IoU=0.50:0.95) medium	0.850	**0.894**
AR (IoU=0.50:0.95) large	1.000	1.000

TABLE 4.1: Performance of RetinaNet vs DETR on the validation set

Both RetinaNet and DETR perform very well on the validation set, consistently giving outputs very close to the hand-labelled data. Overall, RetinaNet marginally outperforms DETR in Table 4.1. Specifically, RetinaNet noticeably outperforms DETR on an IoU threshold of 0.75, but not 0.5. This indicates that DETR detects the presence of objects as reliably as RetinaNet, but the bounding boxes drawn by DETR are on average of slightly more off-centre than with RetinaNet. DETR shows consistently worse performance in detecting small and medium objects. This worse

performance on small objects was something that was noted in the original DETR paper also [4].

4.5.1 Generalisation to synthetic data

Threshold = 0.5	**DETR**	**RetinaNet**
AP (IoU=0.50:0.95)	0.502	**0.602**
AP (IoU=0.50)	0.839	**0.892**
AP (IoU=0.75)	0.472	**0.759**
AP (IoU=0.50:0.95) small	0.518	**0.758**
AP (IoU=0.50:0.95) medium	0.407	**0.601**
AP (IoU=0.50:0.95) large	**0.762**	0.675
AR (IoU=0.50:0.95)	0.542	**0.644**
AR (IoU=0.50:0.95) small	0.603	**0.780**
AR (IoU=0.50:0.95) medium	0.465	**0.645**
AR (IoU=0.50:0.95) large	**0.764**	0.724

TABLE 4.2: Performance of RetinaNet vs DETR synthetic data. Performance is only slightly worse compared to Table 4.1

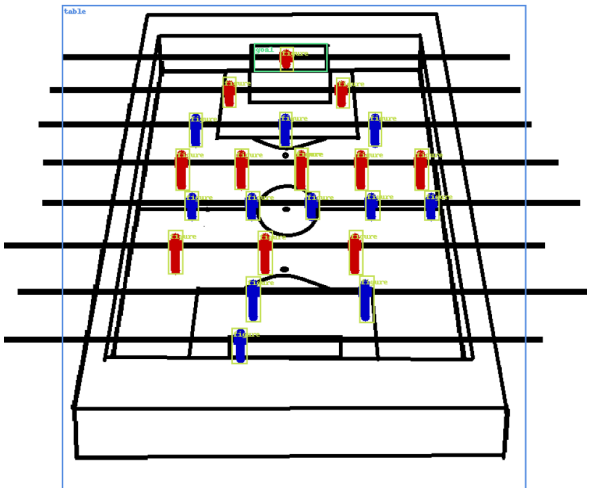


FIGURE 4.7: Example synthetic image labelled by DETR, still showing good object recognition

When tested on the synthetic data introduced in Section 4.2, both models still performed well, giving similar performance to the validation set (Table 4.2). While DETR performs worse in every metric on the synthetic data, RetinaNet performs better on detecting small objects (the figures and ball). While both models still detect figures reliably, giving 83.9% and 89.2% APs for an IoU of 0.5 (Figure 4.7 shows an example of DETR's performance), the clear outlines in the synthetic data (compared to the often blurry figures in the frames taken from YouTube) apparently benefit

RetinaNet more than DETR and allows RetinaNet to draw even more accurate bounding boxes around figures. As both models still show satisfactory performance on the synthetic data for this section of the thesis, the synthetic data was used in subsequent sections to help analyse whether DETR exploits spatial relations.

4.6 DETR decoder queries

In the original DETR paper [4], the authors noted that the decoder queries acted as a learned positional encoding. When analysing how DETR exploits spatial relations, it is important to check how the model may also learn the absolute position of figures. If the model becomes overly reliant on absolute spatial values, any shift in the position of the table in the scene could break the model's performance. Furthermore, DETR could theoretically learn to classify figures based only on their outward appearance and absolute position, forgoing any exploitation of relative spatial regularities. The data augmentations during training are intended to mitigate this trivial reasoning.

Because the different decoder queries are able to attend to each other in the decoder self-attention layers, the queries when trained tend to spread out over the image in the way that would return the best performance on the training data. Figure 4.8 shows the average locations of each query, colour coded by classification [red=figure, magenta=table, red=goal] on a randomly sampled subset of the synthetic data. It seems to be apparent that for the most part, each query seems to have a pattern associated with only one figure object.

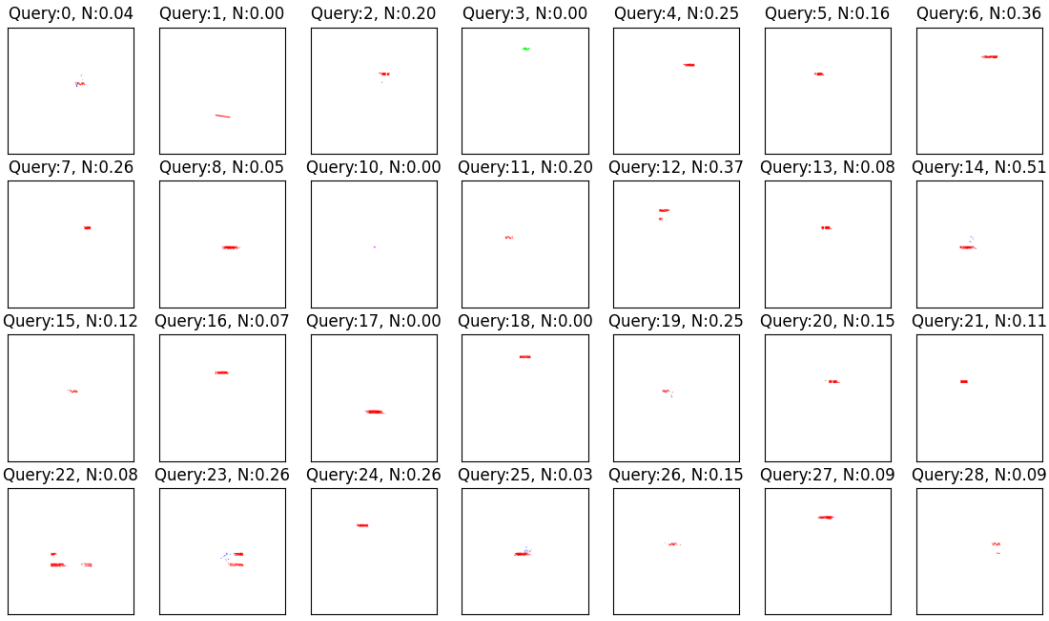


FIGURE 4.8: Query bounding box centres, colour-coded by class [red=figure, green=goal, magenta=table]. Decoder queries are introduced in Section 3.3.3

The parameter **N** in Figure 4.8 is the percentage of classifications related to each query that are returned as the null class, or return a score beneath a threshold of 0.5. Values of **N** are shown again in Table 4.3, where it is apparent that query 9 is mostly to classify the ball, which is non-existent in the synthetic data. Query 9 contributes the blue spots to Figure 4.9, showing that sometimes a ball is detected through this query even when no ball exists. It is also interesting to note that query 10 is used to classify the table object 100% of the time, and query 3 classifies the goal every single time also.

Query	0	1	2	3	4	5	6	7	8	**9**	10	11	12	13	14
% Null	4	0	20	0	25	16	36	26	5	**79**	0	20	37	8	51
Query	15	16	17	18	19	20	21	22	23	24	25	26	27	28	
% Null	12	7	0	0	25	15	11	8	26	26	3	15	9	9	

TABLE 4.3: Percentage of each query that results in a detection of the null class as a detection of low score (synthetic data)

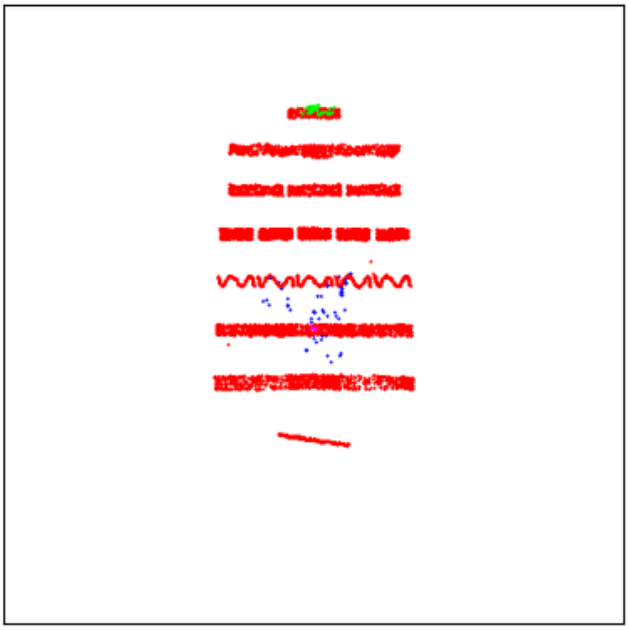


FIGURE 4.9: Queries from Figure 4.8 superimposed on each other into a single Figure, clearly showing the geometry of the foosball table

4.7 Visualising attention

The attention mechanism possessed by DETR is what this thesis hopes to explore with regards to whether or not the transformer model is aware of the surrounding

spatial regularities when detecting each object. This section focuses on visualising this attention and showing how DETR might attend to spatial regularities.

4.7.1 Encoder self-attention

In Figure 4.10, the centres of 6 bounding boxes returned by DETR were selected at random. These centres were then passed as coordinates to index the encoder self-attention map, and therefore return what DETR pays attention to when it classifies an object at a certain location. While the image in Figure 4.10 shows human players on either side of the table, the only discernible shapes in the self-attention maps relate to the foosball table itself. In the top left, top right, and bottom left attention maps in Figure 4.10, hints of individual figure shapes can be seen. In the middle and bottom right attention maps, a darker shape characteristic of a general outline around all the figures appears to be present. The top two attention maps, which both focus on figures, seem to attend also to the very top of the image, which could be that the model has learnt to detect figures close to the end of the table by also paying attention to the end of the table itself.

Beyond this, the encoder attention maps do not suggest much in the way of exploiting spatial relations. For the most part, the attention magnitudes are much higher at the exact location of the sample, and neighbouring spatial regularities are not often emphasised. In the bottom left attention map, it appears as though the self-attention may extend to multiple surrounding figures, but this behaviour is not consistent throughout the self-attention maps.

FIGURE 4.10: Sampling the self-attention maps of DETR's final encoder block at points in the scene corresponding to 6 object centres

4.7.2 Encoder-decoder cross-attention

In the original DETR paper, the cross-attention map of the final decoder is shown to detect objects by attending to their extremities and unique features. It would make sense for DETR to attend to figures nearby to the figure that it detects, as just like you never see a horse without hooves, you rarely see foosball figures without neighbouring figures (either on the same bar or adjacent bars). The encoder-decoder

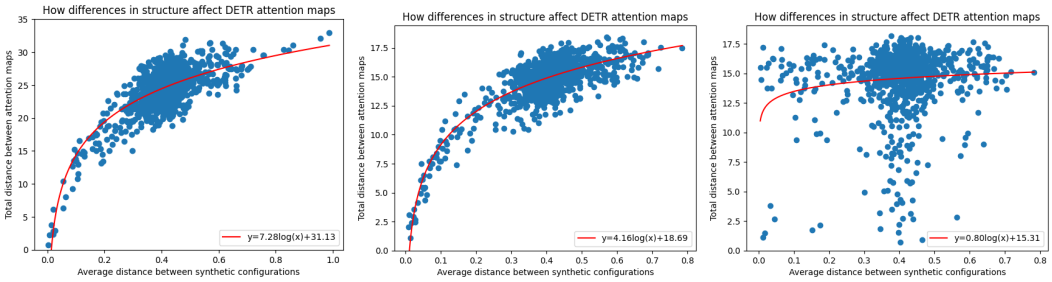
cross-attention map has a shape of $(N_{queries})$x$(H_0 * W_0)$, which can be reshaped to give 29 visual attention maps that can be overlayed on top of the original image. Figure 4.11 shows three examples of this.

(A) (B) (C)

FIGURE 4.11: Three examples of DETR cross-attention, which show recognition of the bars. For clearer visualisation, the magnitudes were multiplied 5x

When detecting a figure object, the cross-attention does not only focus on the object to be detected, but also all the other figures on each bar (Figure 4.11). This is a clear indication that DETR has at the very least perceived the notion of how each figure normally has adjacent figures stretching out in a consistent horizontal arrangement.

To create a point on the scatter plots shown in Figure 4.12, two frames from the synthetic data were randomly sampled. As established in Section 4.2 and Figure 4.4, each synthetic frame is created by a vector of 16 inputs $\in [0, 1]$. The average value of the distance between the input vectors for both frames gives an approximate scalar measure for the degree of difference between the two synthetic frames. Using the behaviour of the queries in Section 4.6, a mostly one-to-one mapping between query and object to be detected can be assumed. Therefore, the sum of the difference between the attention maps associated with both randomly sampled frames provides a scalar representation of the change in the cross-attention between two frames for each object in the scene. Figure 4.12a shows the plots of this difference in the cross-attention maps against the difference in table configuration. As expected, a positive correlation confirms that the cross-attention from DETR is directly affected by the spatial arrangement of the figures in each frame. To specifically target the cross-attention outside of the object being detected, all values in the cross-attention map above 0.1 were set to 0. This thresholded data now only contained the attention to spatial regularities outside of the object to be detected. Figure 4.12b shows the same relationship for this data. Figure 4.12c is a sanity check on this thresholded data, where the two thresholded attention maps of each sampled frame are shuffled before calculating the distance between them.



(A) (B) (C)

FIGURE 4.12: How the magnitude of changes the structure of the table affects DETR's cross-attention maps. There is a clear positive correlation.

4.8 Retraining DETR with positional classes

Although there is some evidence that DETR consistently attends to spatial regularities when detecting figures, this exploitation of spatial relations is not evident in the model's performance on both the validation set (Table 4.1) or the synthetic data (Table 4.2). Intuitively, with only four classes it is still possible to achieve high performance on object detection by relying on feature representations alone. Each class (ball, figure, goal, table) in this scenario has a different appearance from the previous class, which allows them to be detected by outward appearance. Therefore, this section modifies the detection problem to include 11 classes of figure, rather than a single 'figure' class. Table 4.4 and Figure 4.13 show the new classes introduced and how they were positioned in the annotations.

Index	0	1	2	3	4	5	6
Class	Ball	GK	RCB	LCB	RM	RCM	CM
Index	7	8	9	10	11	12	13
Class	LCM	LM	RW	ST	LW	Goal	Table

TABLE 4.4: New positional classes, see Figure 4.13

The positional classes (classes 1 to 11 in Table 4.4) should have a near identical appearance to each other, and only be differentiable by their location on the foosball table. Therefore, if a model wishes to generalise well on these classes, it should learn some form of spatial regularity recognition. The idea is that DETR, with the cross-attention illustrated in Section 4.7.2, will do this more efficiently than RetinaNet, which has no such attention and relies more heavily on image features.

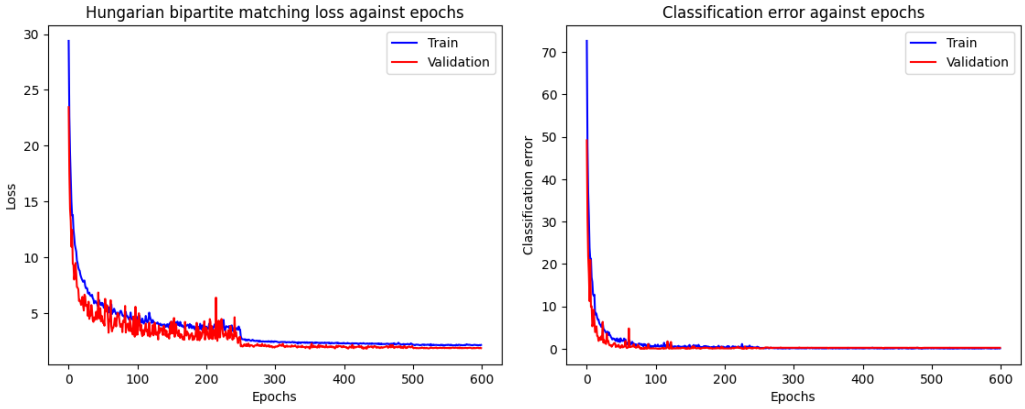
When training DETR and RetinaNet on positional classes, the same train and validation images were used, and only the annotations were modified. Previously, the data augmentations included a horizontal flip, which was removed for the training with positional classes. However, the other augmentations (resizing, cropping) were left in place to discourage DETR from learning trivial reasoning via queries (Section 4.6). The number of epochs was increased from 500 to 600, and the learning rate

FIGURE 4.13: Example of a hand-labelled foosball scene (Figure 4.2) converted to positional labelling

drop occurred every 250 epochs by an order of magnitude of 0.1, starting at a value of 0.0001.

Figure 4.14 shows the training process. Comparing Figure 4.5 and Figure 4.14, the initial loss (a) and class errors (b) start at larger values, especially the class error. However, after 300 epochs the class error settles at 0, something it never does for the original 4 classes during training. One reason for this could be that the positional classes force DETR to learn to exploit spatial regularities, which in turn boost the reliability of the model. However, as will be discussed later in this section, this is probably not the case. An alternative explanation would be that because each class now only contains a maximum of two occurances per image, DETR can more directly associate the output class with the bounding box coordinates. This is true for many one-stage detectors, which can benefit from having a variety of classes as the bounding boxes they output are indirectly associated with their classification output. The lower classification error also presents itself in a higher detection mAP and mAR performance when detecting small objects on the validation set compared to when only four classes were used, and is also observable with RetinaNet (Table 4.5).

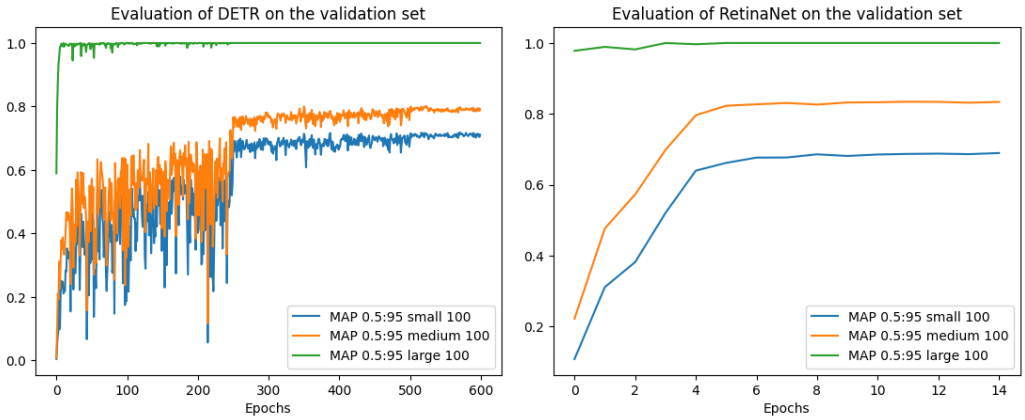
The mAPs on the validation set improve more slowly for both models (Figure 4.15). This is due to the increased number of classes and the similarity in appearance between the classes. However, by the end of training there is a similar split between the



(A) Hungarian loss (Eq 3.2) (B) Class Error

FIGURE 4.14: DETR training curves for the train and validation sets in Section 4.1 annotated as 14 classes, dependent on position

performance on large, medium and small objects to that observed in Figure 4.6. This slow improvement is most noticeable in RetinaNet, where it takes roughly 6 epochs to approach optimal performance, whereas in Figure 4.6b RetinaNet approaches close to optimal performance after 2 epochs. Given that the appearance of many classes in this new problem should be very similar, the good performance of RetinaNet on the validation set comes as a surprise, and suggests that each figure might contain subtle differences in appearance depending on their position.



(A) mAPs in training, DETR (B) mAPs in training, RetinaNet

FIGURE 4.15: Validation set APs for differently sized objects throughout training with positional class annotations, showing slower convergence compared to Figure 4.6

Tables 4.5 and 4.6 show the AP and AR comparisons on the original 4-class annotations with two added columns for the new 14-class positional annotations. In Table 4.5, both DETR and RetinaNet show improved performance on 'AP

(IoU=0.50:0.95) small', compared with the 4-class annotations, meaning that both models are able to detect the figures in the validation set better if they are split up into positional classes rather than kept in a single 'figure' class. However, the overall performance metrics found in the top three rows remain very similar for both sets of annotations on the validation set.

Threshold = 0.5	**DETR**	**RetinaNet**	**DETR**	**RetinaNet**
AP (IoU=0.50:0.95)	0.763	**0.798**	0.741	**0.771**
AP (IoU=0.50)	**0.933**	0.928	**0.917**	0.903
AP (IoU=0.75)	0.820	**0.886**	0.887	**0.899**
AP (IoU=0.50:0.95) small	0.556	**0.628**	0.681	**0.689**
AP (IoU=0.50:0.95) medium	0.818	**0.869**	0.760	**0.834**
AP (IoU=0.50:0.95) large	1.000	1.000	1.000	1.000
AR (IoU=0.50:0.95)	0.798	**0.817**	0.772	**0.801**
AR (IoU=0.50:0.95) small	0.613	**0.663**	0.719	**0.725**
AR (IoU=0.50:0.95) medium	0.850	**0.894**	0.785	**0.865**
AR (IoU=0.50:0.95) large	1.000	1.000	1.000	1.000

TABLE 4.5: Table 4.1 extended to include performance on positional classes. Surprisingly, RetinaNet still outperforms DETR on the validation set with positional annotations

Threshold = 0.5	**DETR**	**RetinaNet**	**DETR**	**RetinaNet**
AP (IoU=0.50:0.95)	0.502	**0.602**	**0.436**	0.197
AP (IoU=0.50)	0.839	**0.892**	**0.743**	0.305
AP (IoU=0.75)	0.472	**0.759**	**0.450**	0.204
AP (IoU=0.50:0.95) small	0.518	**0.758**	**0.372**	0.196
AP (IoU=0.50:0.95) medium	0.407	**0.601**	**0.514**	0.186
AP (IoU=0.50:0.95) large	**0.762**	0.675	**0.800**	0.293
AR (IoU=0.50:0.95)	0.542	**0.644**	**0.545**	0.258
AR (IoU=0.50:0.95) small	0.603	**0.780**	**0.596**	0.258
AR (IoU=0.50:0.95) medium	0.465	**0.645**	**0.581**	0.269
AR (IoU=0.50:0.95) large	**0.764**	0.724	**0.800**	0.313

TABLE 4.6: Table 4.2 extended to include performance on positional classes. RetinaNet suffers a large drop in performance, while DETR is shown to be more robust

Previously on the synthetic data, good generalisation to an unseen table was observed in Section 4.5.1. However, for the new positional annotations this is no longer the case, as both models suffer noticeable decreases in detection performance when implemented on the synthetic data. However, Table 4.6 shows that RetinaNet suffers far worse losses in performance when compared to DETR, in every category. Given that the only difference between the first two columns and the second two columns is a difference in class labelling, RetinaNet shows a clear ineptitude in

detecting figures based mostly on their position in a scene. Table 4.6 suggests that DETR may have learned some spatial patterns pertaining to each positional class.

4.9 Visualising attention for DETR positional classes

Although Table 4.5 shows a better performance for DETR relative to RetinaNet when switching to classes defined mostly by spatial relations, the reason why is still unexplained. Data showing how DETR adapts its detection criteria and attention maps to exploit spatial relations is still needed as confirmation that these results have not come simply from luck or trivial reasoning. Therefore, the same attention experiments as earlier were performed again for the new DETR model.

4.9.1 Encoder self-attention

In the encoder self attention, the same patterns are visible in Figure 4.10 as with 4.16. In Figure 4.16, the middle left and bottom left attention maps show some resemblance of the model attending to other figures and to the sides of the table, but other samples such as the middle right attention map show no such pattern, despite also being sampled at the centrepoint of a figure. The bottom right attention map focuses on sampling the attention map in the middle of the table in both Figure 4.10 and Figure 4.16, and the corresponding attention maps look very similar.

FIGURE 4.16: Sampling the self-attention maps of the final encoder block of 'positional DETR' at points in the scene corresponding to 6 object centres

From a qualitative analysis, the encoder attention once more (for the most part) appears to ignore objects outside of the table, such as the players either side. This is consistent with Figure 4.10, and implies that the encoder attention helps the model interpret table shapes and configurations while ignoring background movements. Beyond this, the varying patterns of the encoder make it difficult to intuitively guess how the encoder chooses its attention magnitudes. In Figure 4.16, the bottom and middle attention maps on the left show contrasting higher and lower attention values when attending to the edges of the table, showing the variance in encoder patterns even when pointing to similar objects.

4.9.2 'Positional DETR' Decoder Queries

Finding 11 distinct colours is quite challenging, so for the visualisation of the positional figure classes, some figures were grouped into overarching classes associated with the bars that they are found on. Table 4.7 shows how the 11 positional classes are grouped into 4 classes, namely the goalkeeper (GK), defenders (DF), midfielders (MD), and forwards (FW).

Position	Encompassed Classes	Colour
Ball	Ball	blue
GK	GK	red
DF	LCB, RCB	yellow
MD	LM, LCM, CM, RCM, RM	black
FW	LW, RW, ST	cyan
Goal	Goal	green
Table	Table	magenta

TABLE 4.7: Grouping positional classes for clearer visualisations. Legend for Figures 4.17 and 4.20. See Figure 4.13 for a labelling including the different 'Encompassed Classes'.

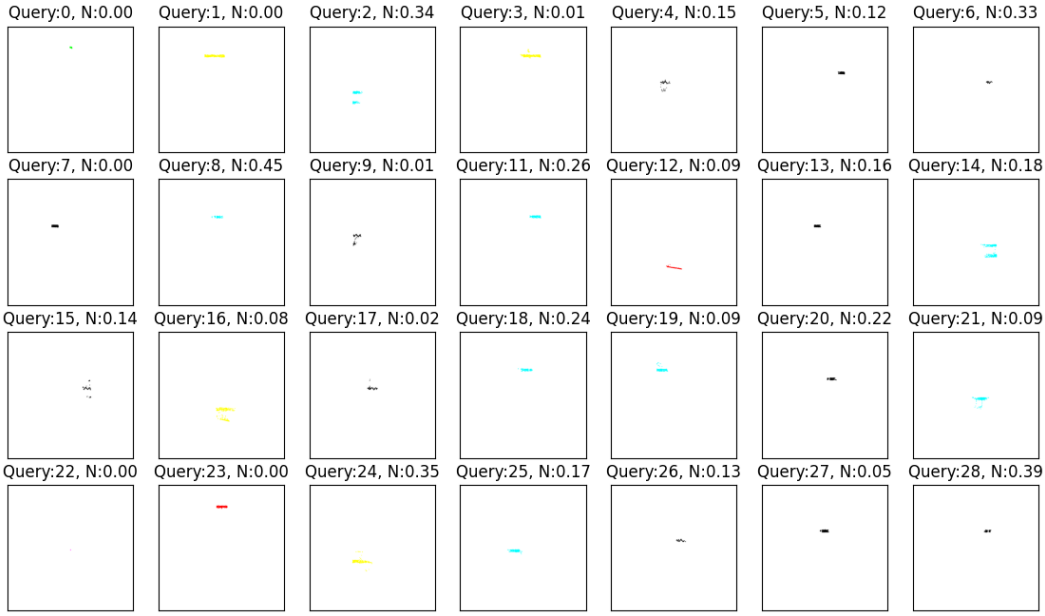


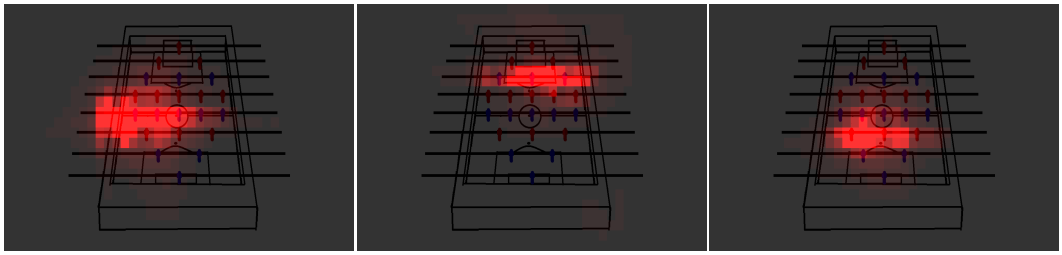
FIGURE 4.17: Query bounding box centres, colour-coded by class (see Table 4.7)

Figure 4.17 shows that each query now strictly only classifies a certain class inside Table 4.7, as each query subfigure shows classifications of only one kind of colour. This fits with the hypothesis that bounding boxes and classes, for one stage detectors, are intrinsically linked and that DETR benefits from not being forced into abstraction when detecting figures. Figure 4.20a shows what these queries look like

when combined into one image. Figure 4.17 shows that query 0 is trained as the goal query, query 22 is trained as the table query, and query 10 is the ball query (not included).

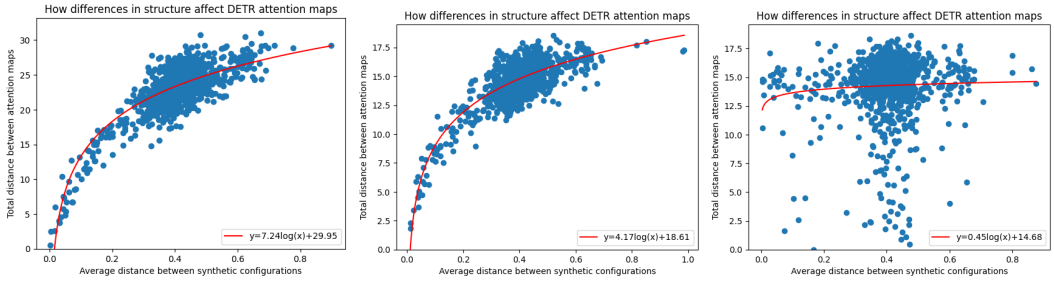
4.9.3 Encoder-decoder cross-attention

Just like in Section 4.7.2, the encoder-decoder cross-attention for 'positional DETR' shows a recognition of other figures on the same bar. Figure 4.18 shows the average attention map for three different queries on the synthetic data. These attention maps have been multiplied by 10, to emphasise small attention values also. All three images show a behaviour of observing along the bar, and at the average position of the object associated with each query (a LM, ST and RW object respectively) there is also some vertical cross-attention. Interestingly, Figure 4.18a shows that when classifying a left midfielder, the model also attends outside the table to make sure that the figure is indeed the left-most figure on the bar.



(A) LM class - Q9 (B) ST class - Q18 (C) RW class - Q25

FIGURE 4.18: The average of all cross-attention maps on 2000 synthetic images, multiplied by 10 for clear visualisation. The cross-attentions span outside of the range of movement of each decoder query's target figure



(A) (B) (C)

FIGURE 4.19: How the magnitude of changes to the structure of the table affects the distance between DETR's attention maps. This shows the same consistent relationship as Figure 4.12.

Compared to DETR trained on 4 classes, the attention maps of positional DETR no not seem to show an increased average magnitude or possess a noticeable change

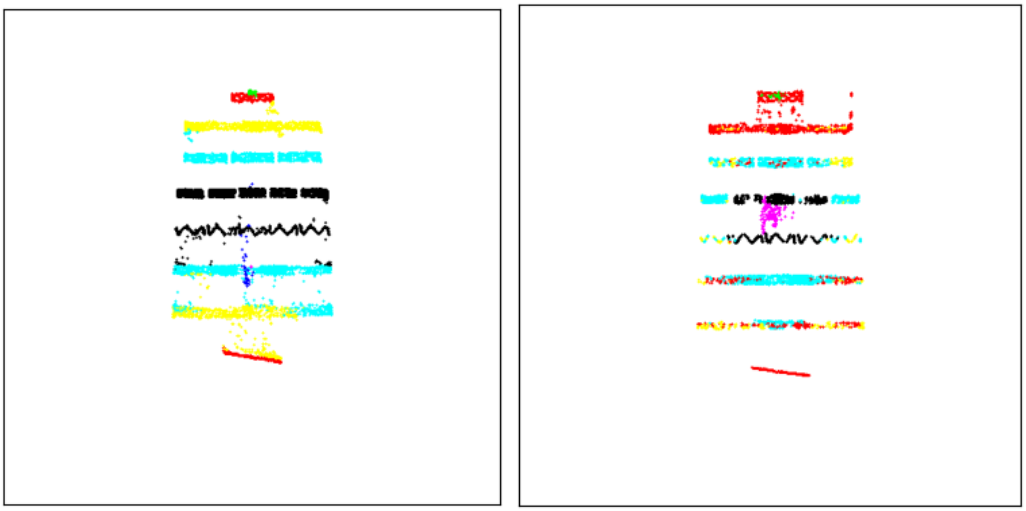
in behaviour. This can be noted in how Figure 4.12 (showing how attention maps vary with structure) and 4.19 are nearly identical. One would expect that if there was a significant difference in the cross-attention maps between the two variants of DETR that the magnitude on the y-axis might shift, or the line of best fit might have a different shape. Instead, all subfigures in both Figures look identical.

4.10 Visualising the performances in Table 4.6

Mentioned earlier in this section were the possible spatial regularities that DETR could learn to exploit based on just the figures of the foosball table:

- Even spaces between figures on each bar, and the immutable number of figures present on every bar
- Even spaces between each bar and its neighbouring bars, and the consistency in there being 8 bars

The question now is exactly which of these spatial relations DETR has learned to exploit, if any, to produce the superior performance in Table 4.6. One can image that if DETR were to learn the first point, and thereby recognise the immutable numbers of figures on each bar, the model would only associate certain positional classes with bars containing a certain number of figures. Therefore, one would expect that DETR could use this logic to separate the classes in Table 4.7 with high accuracy.



(A) Figure 4.17 combined: DETR categorisation of positions

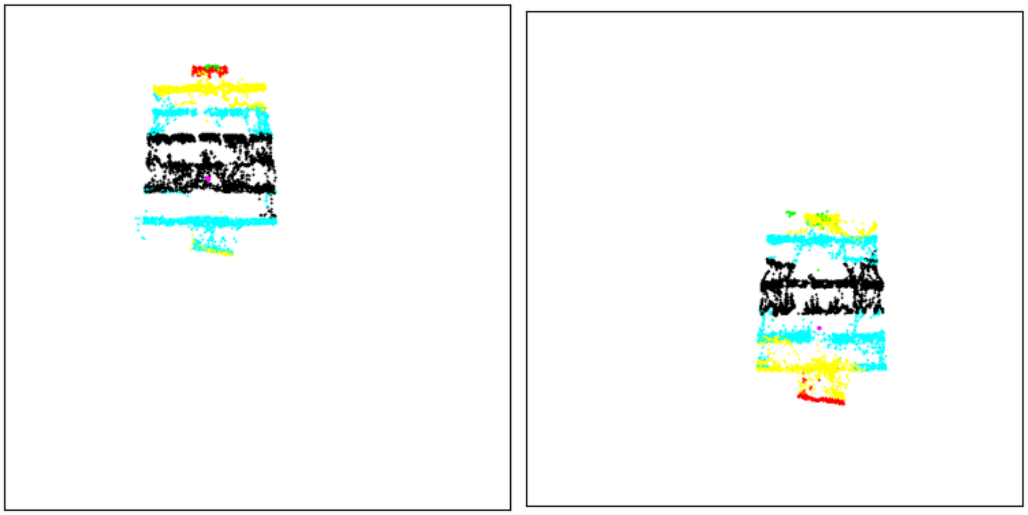
(B) RetinaNet categorisation of positions

FIGURE 4.20: Comparison of DETR and RetinaNet on positional classification. A perfect performance would have each bar in the correct single colour. See Table 4.7 for a legend.

Figure 4.20 shows the relative position of bounding box centres, with the colour being their associated class as per Table 4.7. To create this Figure, 500 random samples were taken (without replacement) from the synthetic data. For each sample, the corresponding output of a model (either DETR or RetinaNet) containing bounding boxes and scores was thresholded to only include scores above 0.5. For both models, the relative centres of the bounding boxes $\in [0, 1]$ were plotted on a 500x500 pixel white image as a coloured dot according to the class label returned by the model.

Figure 4.20 shows strong evidence that DETR has learned to recognise defenders, midfielders and forwards by their positions. In the synthetic data, all figures on the foosball table look exactly the same, and this confuses RetinaNet, which is much more unsure about what positional classes belong on which row (Figure 4.20b).

The question remains, though, as to what extent of DETR's reasoning is not based on relative spatial regularities, but rather a reasoning based on absolute position. Both subfigures in Figure 4.21 were constructed by running the same experiment as in Figure 4.20, but with a padding operation extending the bottom and right of the image (Figure 4.21a) for one experiment, and padding extending the top and left of the image (Figure 4.21b) before input to the model. Both of these Figures show a worse performance in the categorisation of positions, with the positional classes apparently shifted down (Figure 4.21a) or up (Figure 4.21b). Therefore, it is evident that the good performance of DETR over RetinaNet on the synthetic data with positional annotations is owed partially to reasoning on the absolute positions of each figure. However, as DETR still performs better than RetinaNet even in the cases where the image has been padded, there must indeed be some exploitation of spatial regularity.



(A) DETR categorisation of positions padded

(B) DETR categorisation of positions padded, version 2

FIGURE 4.21: Effect of moving the foosball table out of centre. The results show that DETR still relies absolute locations to an extent, but continues to separate classes better than RetinaNet.

I would like to end this section by summarising the observations of DETR on positional classes and to form a succint argument for why DETR performs well. For positional classes, DETR trains each of its decoder input queries to only predict a single class, and also trains that query to only output bounding box centres that appear at a certain position relative to the bounding box centres produced by other decoder input queries (Figure 4.17). DETR adapts to changes in the scene by using the encoder self-attention and encoder-decoder cross attention to generalise to different locations of the table in the scene (Figure 4.21). Looking at previous results, this attention focuses horizontally on how many figures are present on the same bar. Therefore, DETR learns to associate its queries that (for example) output the 'midfielder' class not just with their absolute position, but also by looking laterally at the position of figures on the same bar, and also sometimes to the boundaries of the foosball table and/or longitudinal figure neighbours.

Chapter 5

Conclusion

5.1 Summary

This thesis set out to investigate whether Computer Vision models can recognise objects in a scene by learning the spatial regularities of their surroundings. The DETR object detection model was chosen due to its transformer architecture containing a latent method of learning important image patterns in the attention mechanism. Another model, RetinaNet, was trained alongside DETR for each experiment to provide a benchmark and comparison point. RetinaNet was chosen as it is also a one-stage object detector, does not use the attention mechanism, and is also a model with high-performance on object detection tasks.

The experimental method to evaluate the DETR model started with the preparation of two very different datasets. The dataset used to train and validate came from frames extracted from YouTube videos of recorded foosball games. This data was hand-labelled using Segments.ai. The dataset used to test the models was a synthetic dataset generated using Python libraries such as numpy, matplotlib and opencv. As such, the bounding box and class annotations could be generated automatically.

When trained on four classes, including all the figures under a single umbrella class, both models showed APs of more than 80% on an IoU threshold of 0.5. This gave visual performances on the validation set that were very close to the original hand labellings. Upon evaluation using COCO evaluation metrics, RetinaNet appeared to perform better on small and medium objects.

Next, the attention, notably the self-attention maps of the final encoder block and the cross-attention maps of the final decoder block were visualised and analysed to check whether DETR was showing signs of recognising spatial regularities. It was determined that DETR was also attending to adjacent figures with each figure detection. As the difference in the structural arrangement of players on the table increased, the difference between corresponding cross-attention maps increased. Notably, for very similar table configurations, the difference between the cross-attention maps approached 0, indicating a consistent behaviour of attending to adjacent figures.

To reflect this attendance to spatial regularities in performance, the annotations

were modified to remove the umbrella 'figure' class and replace it with 11 unique classes with differences intended to be wholly dependent on figure positioning. When training the models again, higher class errors were initially observed, which led to a slower plateau in validation set APs. However, in DETR the class error eventually decreased to almost 0, better than with the previous annotations. As RetinaNet also saw improvements in the detection of small objects with the positional classes compared to the umbrella class, it was concluded that this could be because the classification and bounding box outputs of one-stage detectors are indirectly linked.

Both models showed good performance on the validation set with positional classes, but no longer generalised well to the positionally labelled synthetic data. RetinaNet in particular showed poor performance, which was visually confirmed by a colour-coded plot of classification locations. A summary of model performances is shown in Figure 5.1. Although there was no clear difference in the behaviours of the encoder or decoder attention maps with positional labels, it was hypothesised that a combination of trained positional input queries and the lateral attention in the cross-attention maps helps DETR to classify which positional figure classes. Specifically, DETR showed much better performance when asked to associate each positional class with the correct bar on the foosball table. This good performance was worsened with padding of the sythetic images, but there was still some evidence of relative spatial regularities being recognised and exploited.

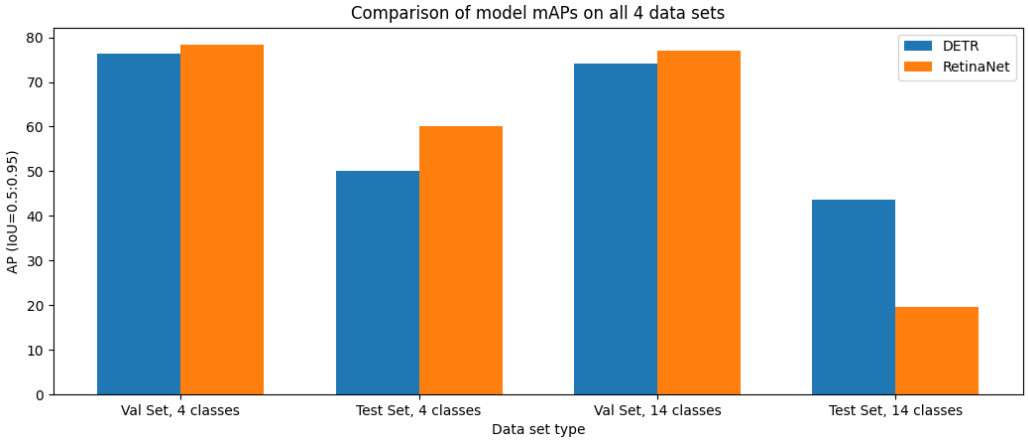


FIGURE 5.1: Overview of model performances on the validation data and the synthetic data with original and positional annotations (Tables 4.5 and 4.6). Outputs were filtered to remove detections with a score lower than 0.5.

5.2 Future works

In future, one suggestion I would make would be to train the DETR model with a wider range of table styles. DETR showed good performance on the validation set for positional classes, but this performance decreased slightly for the synthetic data. This suggests that DETR was able to overfit to the training data, and rather

than exploit spatial relations, was able to associate some characteristic of outward appearance with each positional class. This theory is corroborated by RetinaNet's good performance on the validation set, but inability to generalise to the test set when asked to label figures via position.

For future experiments, I would suggest an ablation study on the DETR model. I noticed during my experiments that DETR and RetinaNet both struggle to classify figures if they do not possess rounded heads. An idea could be to remove the rounded head of a single figure in a bar, and test the average precision of both models on the detection of the figure with the non-rounded head.

When training DETR, the model took very long to converge on good validation set performance. Using Deformable DETR [37] instead of DETR could help speed up convergence, and also help direct the attention maps present in DETR towards attending to spatial relations earlier on in the training process.

Appendices

Appendix A

The First Appendix

A.1 Introduction to Neural Networks

A.1.1 The Perceptron

To introduce this background section, it is important to begin with an explanation of deep learning in the context of neural networks. The core component of artificial neural networks is the perceptron, which is present in all the object detection architectures discussed in this thesis.

At its core, the perceptron is a linear sum of a set of inputs. With a single perceptron, one can construct a function to separate a space with a linear hyperplane. A perceptron typically has the step function to threshold the output of the linear sum corresponding to a class 0 or a class 1, and a single perceptron (also called a neuron) can therefore learn to model the behaviour of logical functions such as AND, or OR, but not XOR. Such a function is known as an activation function, and more complex functions (quadratic, sigmoid or tanh, for example) can create a different boundary shape.

$$out = \phi(\sum_{i=1}^{n}(w_i * x_i) + w_0 * b) \tag{A.1}$$

Equation A.1 uses the notation of w for the weights, ϕ for the activation function and b for the bias term, which is added to have a component of the linear sum independent of the input x. This generation of an output from inputs is called a forward pass. After a forward pass is completed, the weight values of neurons can be trained via back-propagation (also called a backwards pass). A loss function is executed to provide a quantitative measure of how close the predicted output from the forward pass is to the target output. The gradient of this loss function is used to alter the weights of the perceptron based on the value of the loss. Equation A.1 is shown visually in Figure A.1.

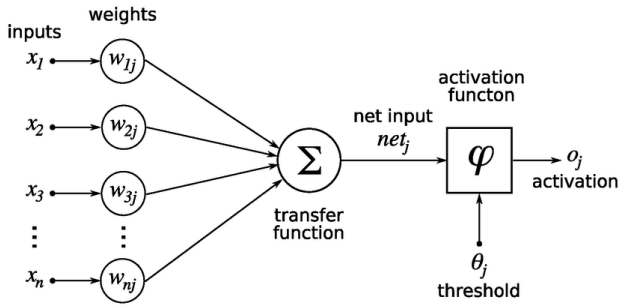


FIGURE A.1: Structure of a single neuron [24]

A.1.2 The Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is a term loosely used to describe a standard feed-forward neural network. An MLP is formed from arranging neurons into 'layers', where the outputs from the neurons from one layer are used as the inputs to the neurons in the subsequent layer.

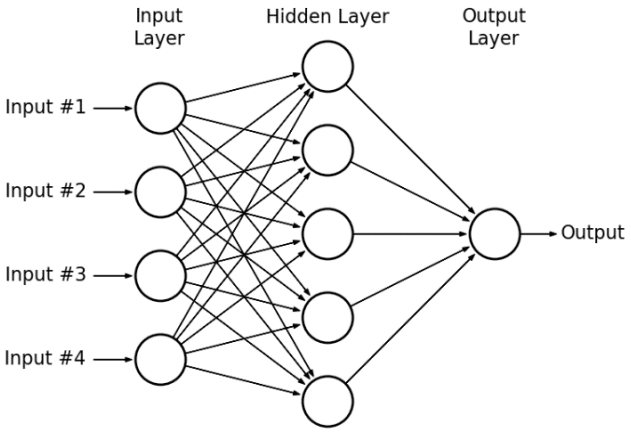


FIGURE A.2: Diagram of an MLP [12]

According to the Universal Approximation theorem, an MLP with a non-constant activation function and only a single hidden layer (an example of which is shown in Figure A.2) is a universal function approximator, provided there is no limit on the number of neurons present in the hidden layer. Because of this, simple MLP architectures can be used in a variety of ways, such as in a control system, where an MLP can take a state representation as input and learn an output voltage to balance an inverted pendulum [22].

MLPs are not only useful for univariate function approximation/prediction. If more output neurons are added, MLPs can be used for classification also. Outputs for classification in MLPs often have the softmax layer as an output, to produce class probabilities between 0 and 1:

$$\sigma(z)_i = \frac{e^{z_i}}{\Sigma_{j=1} e^{z_j}} \tag{A.2}$$

Object detection is a mixture between a classification task, and the approximation of a multivariate function. The network must return both the class of the object detected (classification) and the position of the object in the scene (function approximation). With increased dimensionality on the output, a single-layer MLP could theoretically perform very well on object detection. However, such MLPs are not feasible or efficient because as the complexity of the function approximated increases, the size of the hidden layer increases and (for a fully-connected MLP) the weight matrix therefore quickly becomes unimaginably large.

A.1.3 Deep Neural Networks

When using more than one hidden layer, an architecture can approximate highly complex functions with far fewer total neurons. In this thesis, I define a deep MLP to be an MLP with more than two hidden layers.

In this project, deep MLPs are used inside a larger Transformer architecture (Section 2.1.2). To train these models via backpropagation, a backwards pass through an entire layer follows the principles of gradient descent by exploiting the chain rule (L = loss function, a = activation function, z = weighted summation function, w = weights):

$$\frac{dL}{dw_i} = \frac{dL}{da_i} * \frac{da_i}{dz_i} * \frac{dz_i}{dw_i} \tag{A.3}$$

This enables all weights in a network to be updated after every forward pass. Deep MLPs are more often used than MLPs with a single hidden layer whenever highly complex functions must be approximated. However, they too suffer from high complexity when analysing image data, as the sheer volume of input pixels lead once more to fully-connected layers containing large weight matrices. In the next section, Convolutional Neural Networks (CNNs) are introduced, which have the benefit of modelling only local relations between data rather than being a fully interconnected network. This helps reduce complexity and increase convergence when training a computer vision task.

Bibliography

[1] J. Alammar. The illustrated transformer.

[2] F. Ashfaq, N. Z. Jhanjhi, and N. A. Khan. Badminton player's shot prediction using deep learning. *Lecture Notes in Bioengineering*, page 233–243, 2023.

[3] M. Buric, M. Ivasic-Kos, and M. Pobar. Player tracking in sports videos. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 334–340, 2019.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[6] U. Foos. Unreal foos - trying out garlando table, Mar. 2016.

[7] V. Foosball. Os final: Tokuya kojima vs vincent mok - kaohsiung open 2019 (foosball tournament), Feb. 2020.

[8] FoosballTV. Itsf steinbach 2021 - vorrunde - kuehne, huber vs. lucke, klabunde, Nov. 2021.

[9] Foostorrents. P4p sulzbach 2012 - od finale - schoderer/balic - hansen/grote, Jul. 2012.

[10] K. L. N. L. P. (H02B1a). Exercise session 2 - language modelling, pretraining, self-attention, Apr. 2023.

[11] A. Hashesh. Comprehensive guide to transformers, Apr 2023.

[12] H. Hassan, A. Negm, M. Zahran, and O. Saavedra. Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: Case study el burullus lake. *International Water Technology Journal*, 5, 12 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[14] InsideFoos. Spredeman's wicked pull-kick, Jun. 2013.

[15] InsideFoos. Loffredo's senior moment. the 2015 warrior desert classic senior doubles final., Aug. 2015.

[16] Z. Kalafatić, T. Hrkać, and K. Brkić. Multiple object tracking for football game analysis. In *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 936–941, 2022.

[17] P. Kamble, A. Keskar, and K. Bhurchandi. A deep learning ball tracking system in soccer videos. *Opto-Electronics Review*, 27(1):58–69, 2019.

[18] J. Komorowski, G. Kurzejamski, and G. Sarwas. Footandball: Integrated player and ball detector. *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2020.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018.

[21] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, Z. Wu, D. Zhu, X. Li, N. Qiang, D. Shen, T. Liu, and B. Ge. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models, 2023.

[22] V. Mladenov, G. Tsenov, L. Ekonomou, N. Harkiolakis, and P. Karampelas. Neural network control of an inverted pendulum on a cart. *IEEE*, 8, 01 2009.

[23] Phung and Rhee. A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences*, 9:4500, 10 2019.

[24] S. Ren, G. Chen, T. Li, Q. Chen, and S. Li. A deep learning-based computational algorithm for identifying damage load condition: An artificial intelligence inverse problem solution for failure analysis. *Computer Modeling in Engineering & Sciences*, 117:287–307, 12 2018.

[25] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] segments.ai. Segments.ai: Multi-sensor labeling platform for robotics and autonomous vehicles.

[27] I. TableSoccer. 2014 european champions league replay, Nov. 2014.

[28] I. TableSoccer. 2014 wcs bonzini replay, Jun. 2014.

[29] I. TableSoccer. 2014 wcs garlando replay, Jul. 2014.

[30] I. TableSoccer. Itsf world cup 2014 - final men doubles, Feb. 2014.

[31] truetraders.co.uk. Garden fence privacy screening by true products 3m long.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[33] K. Velumani, R. Lopez-Lozano, S. Madec, W. Guo, J. Gillet, A. Comar, and B. Frederic. Estimates of maize plant density from uav rgb images using faster-rcnn detection model: impact of the spatial resolution, 05 2021.

[34] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee. A survey of modern deep learning based object detection models. *Digital Signal Processing*, 126:103514, 2022.

[35] X. Zhang, T. Zhang, Y. Yang, Z. Wang, and G. Wang. Real-time golf ball detection and tracking based on convolutional neural networks. *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020.

[36] F. Zhu, H. Li, W. Ouyang, N. Yu, and X. Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Mar 2017.

[37] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2021.