**DRDO Project Report: Object Detection**
**Author: Aousnik Gupta**

## 2. Problem Definition

### 2.1. Task Definition

Object detection is a Machine Learning technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class. In this study, we are dealing with the detection of Unmanned Aerial Vehicles (UAVs) in relatively homogeneous backgrounds. The major constraint in this project was to track the source object(s) in real-time. Thus, the objective was to remove many hand-designed components of vision problems like non-max suppression and anchor generation, which not only encodes our prior knowledge of the task, but also increases the computational and time complexities of any vision algorithm. The outline behind fulfilling the objective was to build an object detection model which would approach the problem as a direct set prediction problem; one that would have a global understanding of the target object(s) and correlate them with their surroundings, all at once.

An Object Detection model is expected to output a fixed number of bounding boxes for a given image, along with their corresponding class prediction confidence values (probabilities). Traditionally, Convolutional Neural Network (CNN) based vision models, namely, YOLO (You Only Look Once), Single Shot Detectors and Region based CNNs are employed for this task. However, convolutions have a local understanding of a different parts of an image, i.e., it is able to recognize only the most complex objects in an image. It does so by understanding the simpler patterns in the initial layers of the network and correlating them together in the subsequent ones. However, this correlation extends only to a part of the image, i.e., it has a local understanding and is unable to correlate those complex features of an image which are far away from each other. This leads to unstable confidence values of prediction, i.e., duplicate / multiple bounding boxes for the same object, all of which having nearly the same confidence values, thus, causing overlaps. Therefore, it can be inferred that the model itself has almost no knowledge about the presence of only a single object present at the corresponding area of the image, which calls for the need of Non-Max Suppression (NMS). This algorithm selects the best bounding box, i.e., the one with the highest probability, for a given object and rejects / "suppresses" all the other bounding boxes which have a high overlap (Intersection over Union score) with the "best" bounding box. Generally, an overlap of 0.5 is considered, however, the selection of that hyperparameter certainly affects the final results. Moreover, since each bounding box is compared with every other, the time complexity of NMS becomes $O(n^2)$, thus causing a non-negligible delay in the detection of the target object.

Therefore, in an attempt to approach this problem as a direct set prediction problem and remove NMS and the like, a need to stabilize the confidence of predictions was established. To correlate the local features of an image, a Transformer-based architecture was used, the attention layers of which could leverage the features of an image, returned by a backbone CNN and correlate them together to have a global understanding of where the target object(s) is (/are) and how they correlate with the features (maybe other target objects) throughout the image. The output of the transformer consists of a fixed number of bounding boxes with each bounding box having a confidence of prediction. The overview of the set prediction is depicted in **Fig. 1a**. Since the confidence values are highly stable, the need for hand-designed components is non-existent. The details of the methodology are described in the following section.

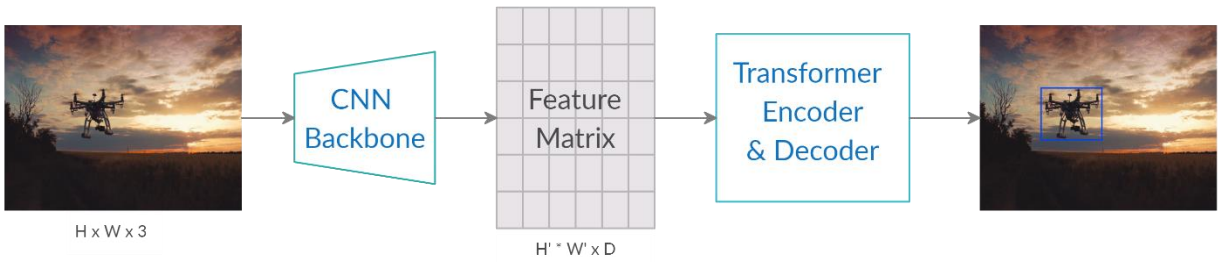


**Fig. 1a.** End-to-End Model Overview

## 2.2. Algorithm Definition

**Fig. 1b.** depicts the detailed architecture of the transformer that uses the image feature matrix returned by the CNN backbone, to correlate the features, globally. The backbone used over here is a pre-trained model of ResNet50, which accepts an input image $x \in \mathbb{R}^{3\times H\times W}$, with 3 color channels and generates a feature matrix $f \in \mathbb{R}^{C\times H\prime\times W\prime}$, where $C = 2048$, $H' = \frac{H}{32}$ and $W' = \frac{W}{32}$. This feature matrix is flattened to $z \in \mathbb{R}^{C\times H\prime * W\prime}$, which serves as the input to the transformer. We may optionally choose to reduce the channel dimension of $f$ from $C$ to a lower dimension $D$, by passing it through a 1x1 convolution layer. Moreover, since the encoder expects a sequence as input, $f$ is collapsed into $z \in \mathbb{R}^{D\times H\prime * W\prime}$. The output of the Transformer consists of a fixed number ($N$) of bounding boxes along with their corresponding classes, i.e., 2 matrices: class predictions $c \in [0,1]^{N\times 1}$, and normalized bounding box coordinates $b \in [0,1]^{N\times 4}$.

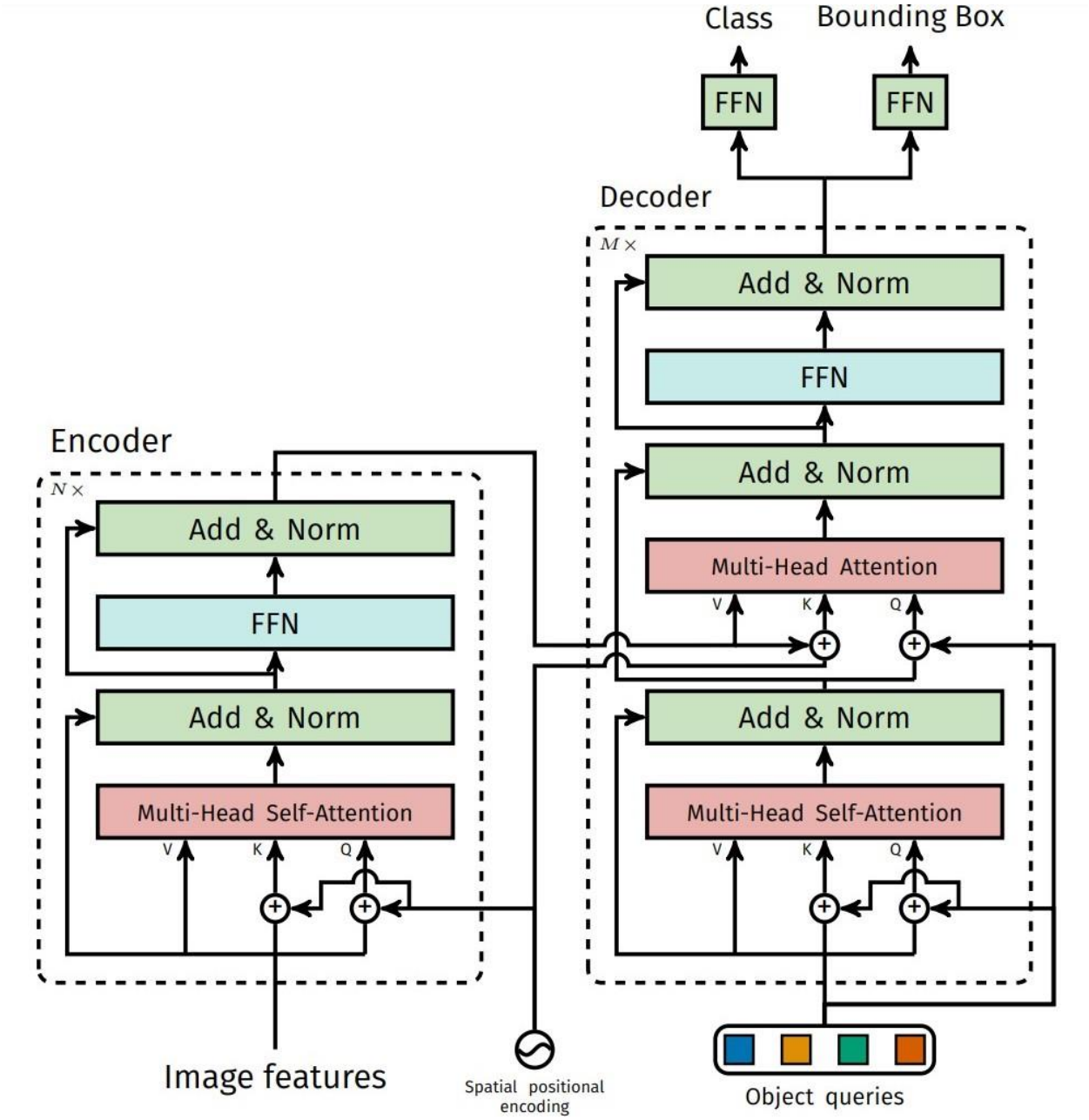


**Fig. 1b.** Architecture of Detection Transformer (DETR)

**Encoder:** Each encoder layer has the standard architecture consisting a multi-head self-attention module and a feed forward network (FFN). Since the transformer architecture is permutation-invariant, fixed positional encodings are supplemented to it at the input of each attention layer. The encoder self-attention layers correlate the features globally with every other feature using pair-wise relations, thus understanding the dynamics throughout the image.

**Decoder:** Each decoder layer has the standard architecture of a transformer, consisting of multi-head self and encoder-decoder attention units, that transform the *N* embeddings of size d along with the encoder's correlation of features into the required output. The conceptual difference with the Transformer used for Natural Language Processing (NLP) and the one used here is that the model yields the probabilities and bounding boxes at once, unlike the traditional transformer which is an autoregressive model, which predicts the output sequence one value at a time. The decoder is also permutation-invariant; hence, *N* input embeddings needs to be different to produce different results. These input embeddings are positional encodings, referred to as object queries, which instead of being fixed, are learnt while training. Similar to the encoder, object queries are added to the input of each attention layer of the decoder. The *N* object queries are then transformed into an output embedding by the decoder, which are independently decoded into box coordinates and class labels by a feed forward network, resulting in *N* number of final predictions. Using self-attention and encoder-decoder attention over these embeddings, the model globally reasons about all objects together using pair-wise relations, correlate object queries to the bounding boxes and minimize the duplication of predictions, thus, being able to use the whole image as context.

**Feed Forward Network:** It consists of 3 separate hidden layers with ReLU activation functions, the first one (optional) having a dimension of *d*, the second, having a hidden dimension of *1* which produces the confidence values and the third with a hidden dimension *4*, which produces the bounding boxes.

**Set Prediction Loss:** The major highlight besides using a Transformer model is the use of the **Hungarian Matching** algorithm. Since the number of predictions is fixed and permutation invariant, there exists a difficulty to score the predictions with respect to the ground truth bounding boxes. The Hungarian Matching algorithm overcomes these difficulties by finding an optimal bipartite matching between the predictions and the ground truth.

To find an optimal bipartite matching, we need to find the optimal permutation, $\hat{\sigma}$ between the two sets, the predictions, $\hat{y}$ and the ground truth, $y$ such that $\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$ is minimum, i.e.,

$$\hat{\sigma} = \arg min \sum_{i=1}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}), \qquad \dots (1)$$

where, $\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$ is the pair-wise matching cost between the ground truth and a prediction at the index $\sigma(i)$, and $y_i = (c_i, b_i)$, where, $c_i$ is the class of the object, $b_i$ is the bounding box coordinates and $\hat{p}_{\sigma(i)}(c_i)$ is the class confidence prediction value at the $i^{\text{th}}$ index. For out purposes, we have 2 classes of prediction namely, background, denoted by $\emptyset$ and UAVs. The matching cost takes into consideration the class confidence values as well as the bounding box coordinates as,

$$\mathcal{L}_{match}\left(y_i, \hat{y}_{\sigma(i)}\right) = -1_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + 1_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}\left(b_i, \hat{b}_{\sigma(i)}\right), \qquad \dots (2)$$

where, the background class is not considered during the calculations and, $\mathcal{L}_{box}\left(b_i, \hat{b}_{\sigma(i)}\right)$ scores the bounding boxes which is a combination of $L_1$ loss and Generalized Intersection over Union loss, and is given by,

$$\mathcal{L}_{box}\left(b_i, \hat{b}_{\sigma(i)}\right) = \lambda_{L1} ||b_i - \hat{b}_{\sigma(i)}||_1 + \lambda_{giou} \mathcal{L}_{GIoU}\left(b_i, \hat{b}_{\sigma(i)}\right), \qquad \dots (3)$$

where, $\lambda_{L1}, \lambda_{giou} \in \mathbb{R}$ are hyperparameters and, $\mathcal{L}_{GIoU}\left(b_i, \hat{b}_{\sigma(i)}\right)$ can be calculated using the **Algorithm. 1**.

This process of finding a one-one matching for direct set prediction makes sure that there are no duplicates present in the predictions, which stabilizes the output probabilities with continued training. After the optimal permutation, $\hat{\sigma}$ is found, the Hungarian Loss is given by,

$$\mathcal{L}_{Hungarian}(y, \hat{y}) = \sum_{i=1}^{N} [-log \hat{p}_{\hat{\sigma}(i)}(c_i) + 1_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)})] \qquad \dots (4)$$

**Algorithm. 1.** Calculation of IoU and GIoU given predicted and ground truth bounding boxes

**input :** Predicted $B^p$ and ground truth $B^g$ bounding box coordinates:
$B^p = (x_1^p, y_1^p, x_2^p, y_2^p), \quad B^g = (x_1^g, y_1^g, x_2^g, y_2^g).$

**output:** $\mathcal{L}_{IoU}, \mathcal{L}_{GIoU}$.

For the predicted box $B^p$, ensuring $x_2^p > x_1^p$ and $y_2^p > y_1^p$:
$\hat{x}_1^p = \min(x_1^p, x_2^p), \quad \hat{x}_2^p = \max(x_1^p, x_2^p),$
$\hat{y}_1^p = \min(y_1^p, y_2^p), \quad \hat{y}_2^p = \max(y_1^p, y_2^p).$

Calculating area of $B^g$: $A^g = (x_2^g - x_1^g) \times (y_2^g - y_1^g)$.

Calculating area of $B^p$: $A^p = (\hat{x}_2^p - \hat{x}_1^p) \times (\hat{y}_2^p - \hat{y}_1^p)$.

Calculating intersection $\mathcal{I}$ between $B^p$ and $B^g$:
$x_1^{\mathcal{I}} = \max(\hat{x}_1^p, x_1^g), \quad x_2^{\mathcal{I}} = \min(\hat{x}_2^p, x_2^g),$
$y_1^{\mathcal{I}} = \max(\hat{y}_1^p, y_1^g), \quad y_2^{\mathcal{I}} = \min(\hat{y}_2^p, y_2^g),$
$$\mathcal{I} = \begin{cases} (x_2^{\mathcal{I}} - x_1^{\mathcal{I}}) \times (y_2^{\mathcal{I}} - y_1^{\mathcal{I}}) & \text{if} \quad x_2^{\mathcal{I}} > x_1^{\mathcal{I}}, y_2^{\mathcal{I}} > y_1^{\mathcal{I}} \\ 0 & \text{otherwise.} \end{cases}$$

Finding the coordinate of smallest enclosing box $B^c$:
$x_1^c = \min(\hat{x}_1^p, x_1^g), \quad x_2^c = \max(\hat{x}_2^p, x_2^g),$
$y_1^c = \min(\hat{y}_1^p, y_1^g), \quad y_2^c = \max(\hat{y}_2^p, y_2^g).$

Calculating area of $B^c$: $A^c = (x_2^c - x_1^c) \times (y_2^c - y_1^c)$.

$IoU = \dfrac{\mathcal{I}}{\mathcal{U}}$, where $\mathcal{U} = A^p + A^g - \mathcal{I}$.

$GIoU = IoU - \dfrac{A^c - \mathcal{U}}{A^c}$.

$\mathcal{L}_{IoU} = 1 - IoU, \quad \mathcal{L}_{GIoU} = 1 - GIoU.$

## 3. Experimental Evaluation

### 3.1. Methodology

In this section we take a look at how the algorithm explained in the preceding section is scored based on its accuracy and speed, the technical details behind its training and, the dataset used to train the vision model.

**Dataset:** We perform the experiments on the Real-World Object Detection Dataset for Quadcopter Unmanned Aerial Vehicle (UAV) [reference_no] consisting of 51.4k training and 5.3k validation 640x480 RGB images. The dataset presents drones in different types, sizes, scales, positions, environments, times-of-day with corresponding XML labels, with up to 12 instances in a single image. To score and compare the accuracy of the model over a batch of images Mean Average Precision (mAP) metric has been used over the validation set.

**Training Parameters:** For the backbone, a ResNet50 network pre-trained on ImageNet database was used with frozen Batch Normalization layers. To keep the math easier, every image from the dataset was resized to 512x512, followed by standard image augmentations such as, random crops to improve training performance. The model has been trained with Adam Optimizer with the initial learning rate set to $10^{-4}$. All the weights and biases had been initialized with Xavier initialization. The model was trained for 100 epochs, with an early stopping criteria of increasing validation loss and decreasing training loss consistently over successive epochs, signifying overfit.

### 3.2 Result Analysis and Comparative Study

This model was compared with the previous most recent Object Detection Architecture by Google Research's state of the art, EfficientDet-D1 [reference_no] and Faster RCNN, in terms of accuracy, using Mean Average Precision score and, latency/FPS. The values of the comparison have been depicted by **Table. 1**. From the table it can be inferred that the removal of hand-designed components, and stabilization of the confidence of prediction, the latency has decreased by a maximum of 68.75% causing up to a 220% increase in observable speed. Though the model comes

with higher number of parameters, its performance with respect to previously SoTA architectures is certainly an improvement towards real-time Object Detection, in general.

| Model | Parameters | FPS (GTX 1650) | Latency (GTX 1650) | mAP |
|---|---|---|---|---|
| Detection Transformer | 41M | 16 | 62.5ms | 0.87 |
| Faster RCNN-FPN | 42M | 9 | 111.1ms | 0.81 |
| EfficientDet-D1 | 6.6M | 5 | 200.0ms | 0.80 |

**Table. 1.** Comparative study

**Few Comparable Outputs:**

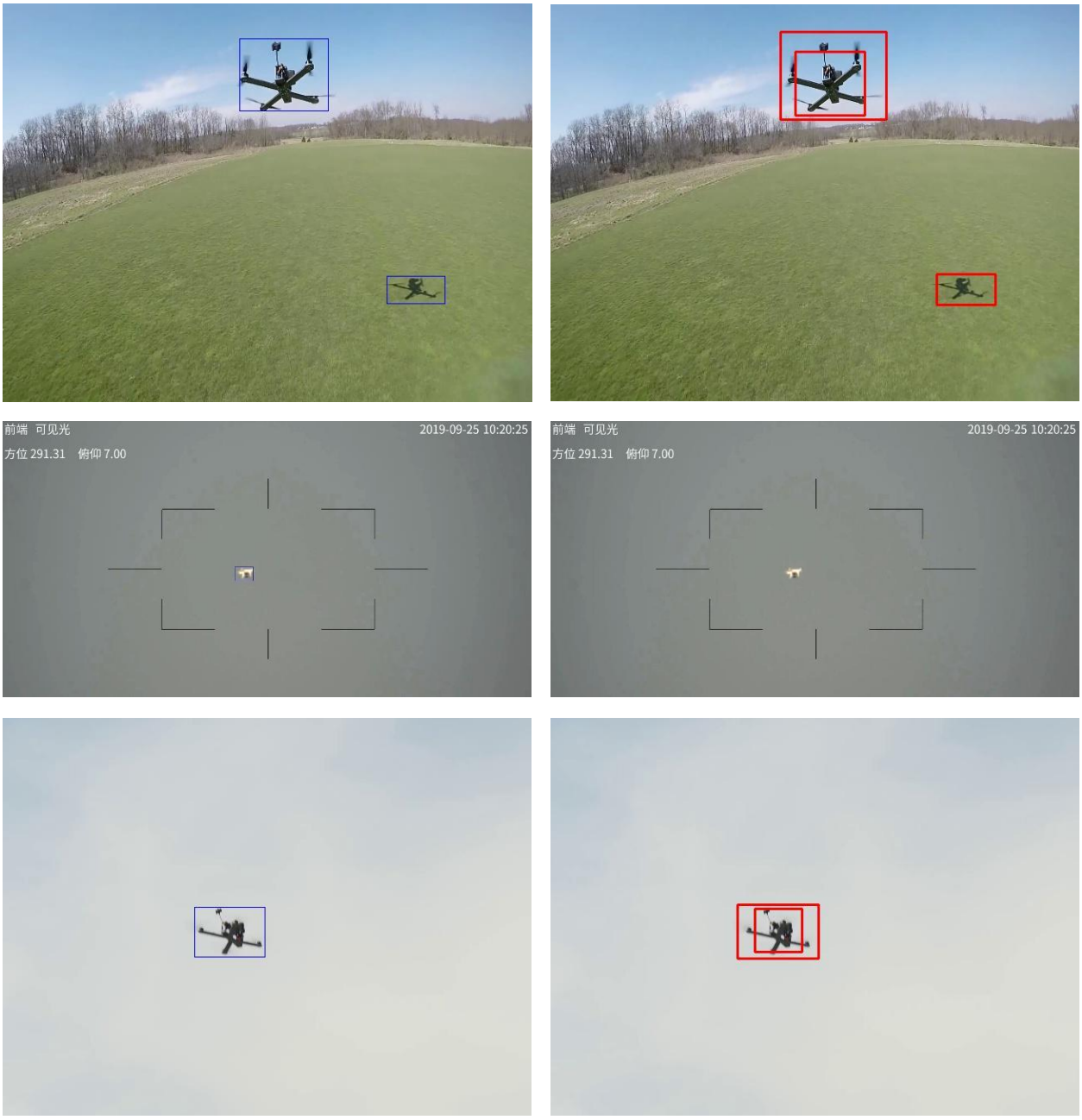

**Fig. 1c.** Comparisons of predictions between the Detection Transformer and EfficientDet-D1