Kendrick Xie
Jei Ho

Introduction:

In this final project, we fine-tuned a pre-trained DETR model to make object detections specifically on soccer games. The dataset we used consisted of 217 images where the training set had 204, and the test set had 13. The dataset includes a wide range of soccer game scenarios including different players, game phases, and lighting conditions. The dataset is labeled with bounding boxes for 4 object classes: balls, goalkeepers, players, and referees. In the end, we found that the model (epochs=15, IoU threshold>=0.5) was able to accurately draw bounding boxes around the detected objects.

Data Preprocessing:

The soccer dataset labels bounding boxes with the form [xmin, ymin, xmax, ymax], while DETR uses the COCO format of [xmin, ymin, width, height]. To match the COCO format, we created a new width and height column for the soccer dataset. Since the original labels include a row for each individual bounding box, we also create a new table with one image per row with a column for file names and a column for the number of bounding boxes in the image.

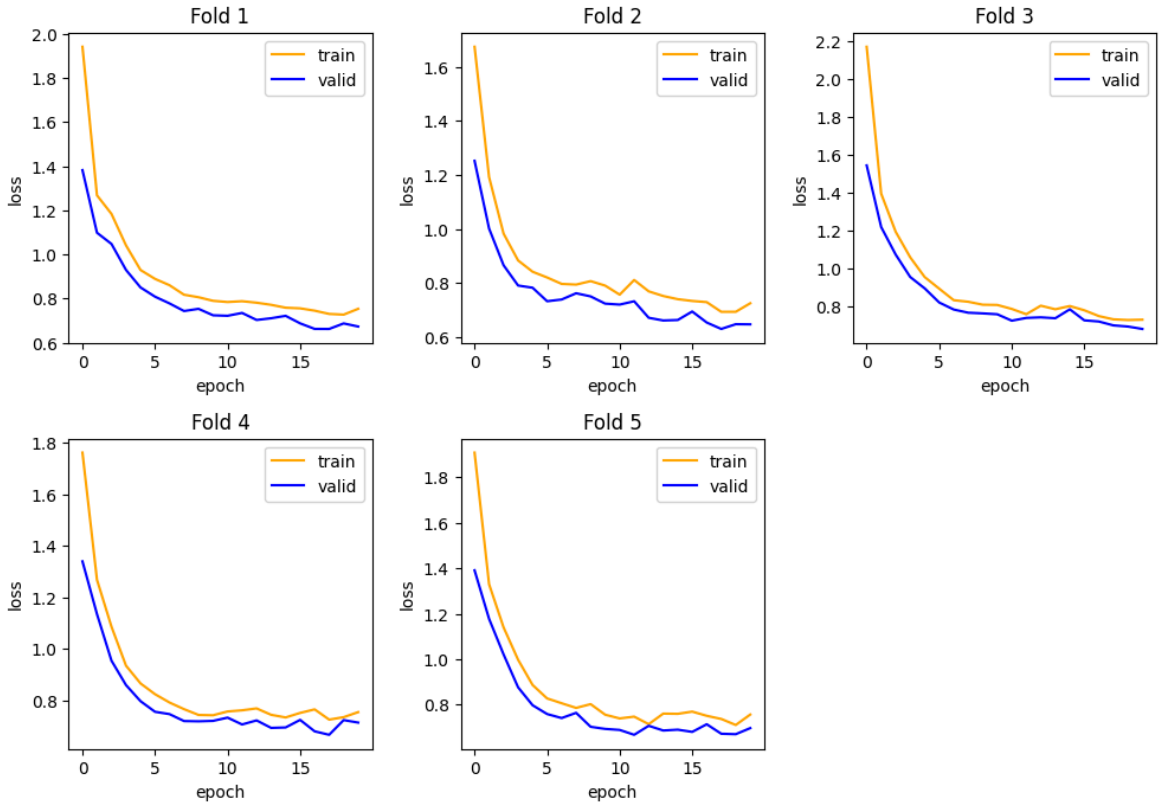
Before training we also transform the images. All images were resized from 1920x1080 to 960x540 to increase efficiency. The hue, saturation, value, brightest, and contrast of training images were randomly shifted a small amount with 0.0 probability. Training images were converted to greyscale with a probability of 0.01. Training images flipped horizontally and vertically, both with a probability of 0.5. Lastly, cutouts were randomly applied to the training images. These modifications of the training data improve model generalization by artificially increasing the diversity of the data.

Defining the Model:

We use a pre-trained DETR model, but modify the parameters to be compatible with 5 classes (balls, goalkeepers, players, referees, and the background) and the max number of bounding boxes found in any of the training images.

Training:
To calculate loss during training and validation, we used an average of classification loss, bounding box loss, and generalized intersection over union loss.

We perform k-fold cross validation with 5 folds for 20 epochs:

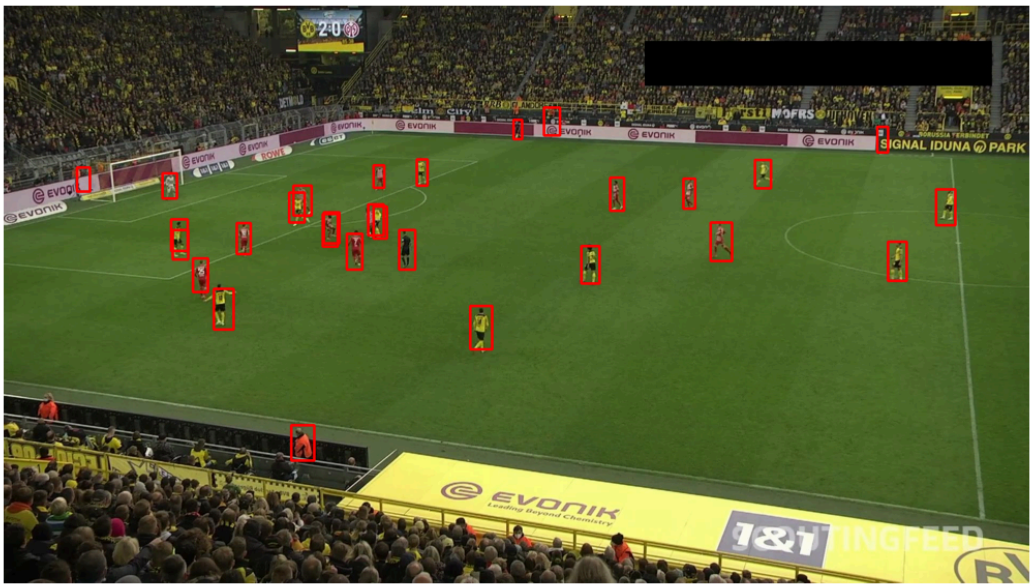


From observing the trends of both training and validation loss, we chose to train over all the training data for 15 epochs to obtain our final model. This gave a final loss of 0.7346.

Results:

Example ground truth bounding boxes:

Example predicted bounding boxes:



We see that the bounding boxes predicted by the model come fairly close to the original ones and it is rare for the model to overlook one of the labeled objects. However, the model does sometimes create bounding boxes at areas where there are no objects and create overlapping bounding boxes on one single object. Such occurrences would be classified as False Positive as the IoU would be low and 0 in some cases where the intersection does not exist between the ground truth and predicted bounding boxes. Notably, the model sometimes identifies people standing off of the soccer field as one of the labeled objects, which would increase the number of false positives. It would be challenging to adjust the model to avoid these false positives as these people can appear very similar to players of the field, especially given the small number of pixels a human occupies in the dataset images relative to the 1920x1080 resolution. This helps explain why the recall score per image is relatively higher than the precision score per image as shown in the table below.

Test Images	Precision (TP/TP+FP)	Recall (TP/TP+FN)
1	0.6666666666666666	0.8333333333333334
2	0.5517241379310345	0.6666666666666666
3	0.5714285714285714	0.6956521739130435
4	0.5862068965517241	0.68

5	0.6896551724137931	0.8333333333333334
6	0.7272727272727273	0.6666666666666666
7	0.7037037037037037	0.7916666666666666
8	0.6785714285714286	0.7916666666666666
9	0.64	0.6666666666666666
10	0.5769230769230769	0.7142857142857143
11	0.5357142857142857	0.625
12	0.8095238095238095	0.68
13	0.6923076923076923	0.75
Average	0.6484383206929626	0.722687529861443

Encoder-Decoder Multi-Head Attention Weights:
Below we visualize the attention weights of the last decoder layer. From the visualization below, we can see the model mainly looks at a small area around detected objects to make predictions.