Computer Vision

Assignment 2

Omer Kamal Ali Ebead

Problem 1:

- I imported the dataset from sklearn, resize the images to only include the important parts.

- Then I split the data into train and test (80% , 20%).
- After that I added a noise of 10% to the training images (whose maximum pixel value is 1 so gaussian noise with mean zero and variance 1) . Below are samples of noisy training images.

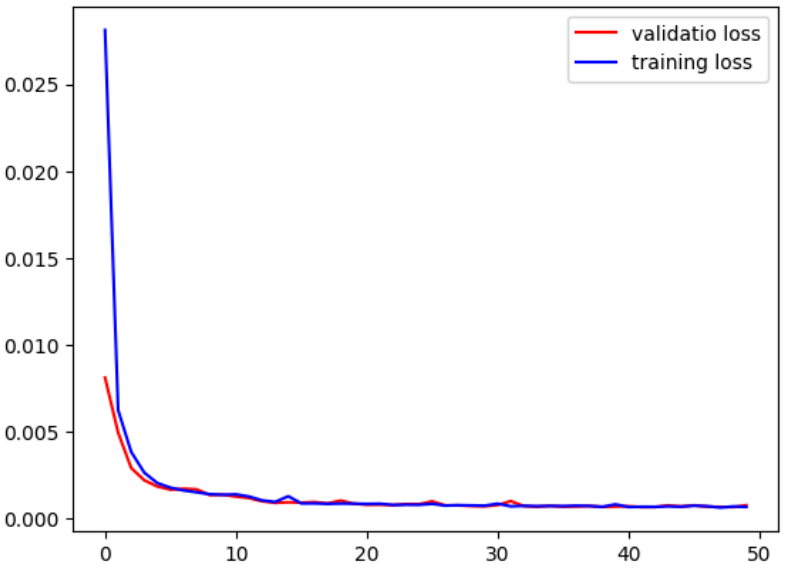
- Then I started to build the encoder, keras has a great tutorial in building autoencoders (https://keras.io/examples/vision/autoencoder/) . Below is the summary of my autoencoder.

Model: "functional_6"

Layer (type)	Output Shape	Param #
input_layer_7 (InputLayer)	(None, 128, 128, 3)	0
conv2d_51 (Conv2D)	(None, 128, 128, 32)	896
average_pooling2d_15 (AveragePooling2D)	(None, 64, 64, 32)	0
conv2d_52 (Conv2D)	(None, 64, 64, 32)	9,248
conv2d_53 (Conv2D)	(None, 64, 64, 32)	9,248
average_pooling2d_16 (AveragePooling2D)	(None, 32, 32, 32)	0
conv2d_54 (Conv2D)	(None, 32, 32, 32)	9,248
up_sampling2d_23 (UpSampling2D)	(None, 64, 64, 32)	0
conv2d_55 (Conv2D)	(None, 64, 64, 32)	9,248
conv2d_56 (Conv2D)	(None, 64, 64, 32)	9,248
up_sampling2d_24 (UpSampling2D)	(None, 128, 128, 32)	0
conv2d_57 (Conv2D)	(None, 128, 128, 32)	9,248
conv2d_58 (Conv2D)	(None, 128, 128, 32)	9,248
conv2d_59 (Conv2D)	(None, 128, 128, 3)	867

Total params: 66,499 (259.76 KB)
Trainable params: 66,499 (259.76 KB)
Non-trainable params: 0 (0.00 B)

- I used mean square error (MSE) as a loss function and Adam as an optimizer.
- I trained the model in 50 epochs and took the first 350 images in the noisy train data as validation data.
- **Results** :



- Then I tested the model in the noisy test set :

```
test = autoencoder.evaluate(x_test_noisy,x_test,batch_size=64)
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 2s 128ms/step - accuracy: 0.9602 - loss: 7.4194e-04
```

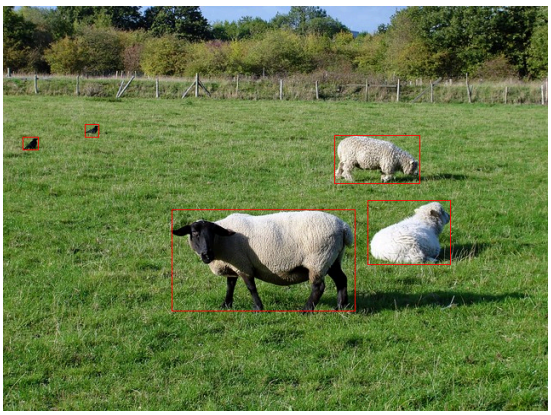
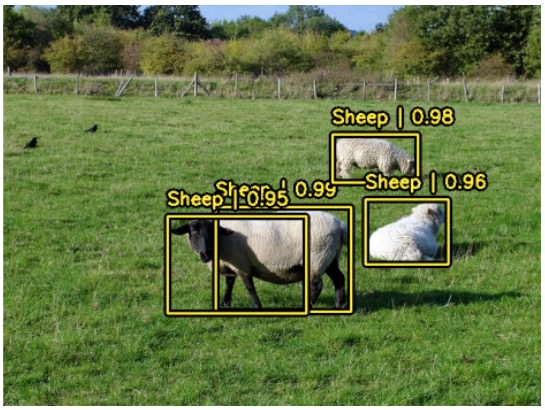
- Lastly I tried to visualise some of the test set outputs to see the prediction of the model.
- 

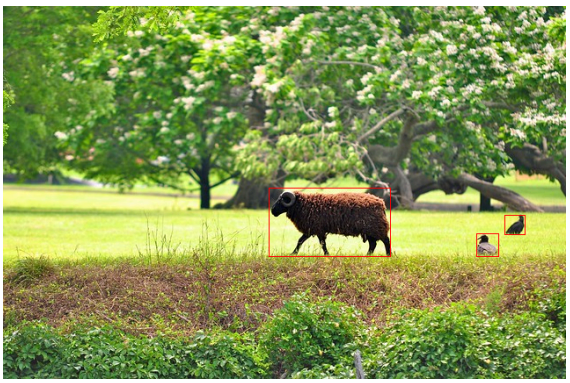
Problem 2:

I chose YOLO v8

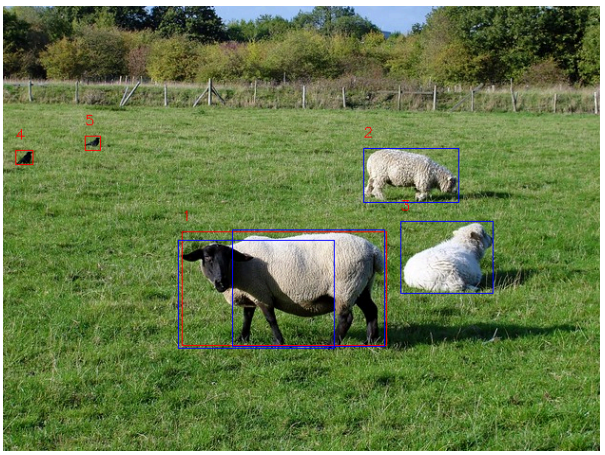
Test images :

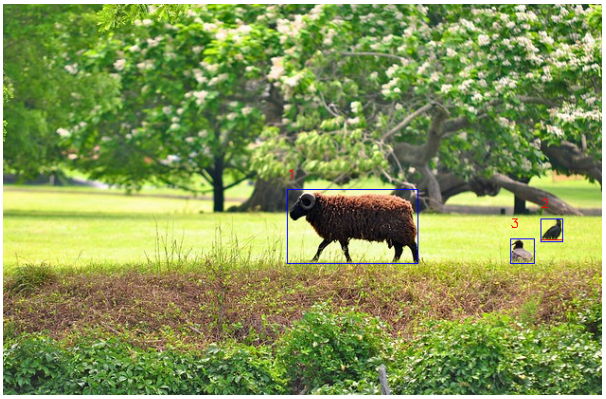
- Then I made the ground truth bounding boxes using open cv.
- Red boxes are target boxes and yellow are the model predictions.




- Mean Average Precision calculation :
    - Precision : what fraction of boxes we made correctly in the boxes we predicted.
    - Recall : what fraction of boxes we correctly detected of all the target boxes.
    - Average Precision : we use it to compare between models as it measures the area between the recall and precision for each threshold.
    - Mean Average Precision: the average precision for all classes .
- We should calculate the precision and recalls for the test images.
- We will be dealing with threshold 0.5@0.05@0.95 (we start at threshold 0.5 and end with 0.95 and add 0.05 every step)
- First I calculated the Intersection over Union for all images for all classes.
- “ for the images below, red is the target bounding box and blue is the predicted bounding box”
- 

- 

- 

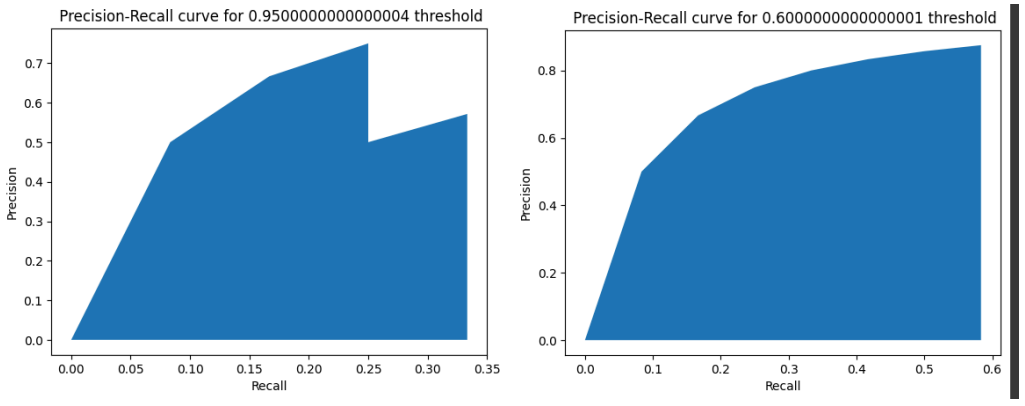
- I created a function to calculate the edges of the box to help me calculate IOU => given x,y,w,h , the edge of the box is:
    - Top left x = x - (0.5 * w)
    - Top left y = y - (0.5 * h)
    - Bottom right x = x + (0.5 * w)
    - Bottom right y = y + (0.5 * h)
- I created a function to calculate the intersections area, which can be represented as the area of the box between the boxes (if exist) :
    - Top left x = max(top left x1, top left x2)
    - Top left y = max(top left y1, top left y2)
    - Bottom right x = min(bottom right x1, bottom right x2)
    - Bottom right y = min(bottom right y1, bottom right y2)
    - Then i checked if there an intersection box or not (if not return 0, if there is, i return the area)
- I created a function to calculate the union which is the area of the 2 boxes minus the intersection between them (if exist)
- Then IOU function will be calling the intersection function and the union function and return the division of them
- After that I made a data frame for all the images which contains most of the information needed to calculate the mAP.

	Image number	target bbox	target class	iou	predicted bbox	predicted class	confidence score
0	image 0	[228, 194, 70, 43]	Sheep	0.915433	[228.24442, 194.75839, 73.878784, 43.30397]	Sheep	0.942403
1	image 0	[448, 192, 89, 48]	Sheep	0.955063	[446.77637, 191.76418, 90.38379, 48.910934]	Sheep	0.921041
2	image 0	[295, 184, 96, 51]	Sheep	0.879498	[290.07745, 184.21776, 108.23135, 51.41365]	Sheep	0.901004
3	image 0	[337, 170, 31, 21]	Bird	0.892493	[337.46216, 170.9074, 31.930634, 21.825134]	Sheep	0.882747
0	image 1	[198, 241, 215, 120]	Sheep	0.547604	[251.14699, 239.96185, 160.39119, 123.63736]	Sheep	0.990598
1	image 1	[389, 153, 100, 57]	Sheep	0.968898	[389.9533, 153.12675, 100.51538, 57.72647]	Sheep	0.979615
2	image 1	[428, 230, 97, 76]	Sheep	0.969189	[428.40094, 230.8805, 97.80505, 76.477875]	Sheep	0.963800
3	image 1	[198, 241, 215, 120]	Sheep	0.656665	[194.53662, 250.55005, 164.0271, 114.80838]	Sheep	0.947794
0	image 2	[302, 207, 138, 78]	Sheep	0.963164	[302.9632, 207.92993, 138.65057, 78.63748]	Sheep	0.983868
1	image 2	[570, 238, 23, 22]	Bird	0.874526	[570.1732, 238.63919, 23.51825, 24.602142]	Bird	0.851862
2	image 2	[538, 259, 25, 26]	Bird	0.932695	[538.0018, 259.70438, 25.405945, 26.279724]	Bird	0.759863

- Sorted it by the confidence score and start the calculation of precisions and recalls.
- Precision :
    - The total number of true positives divided by the the total number of true positive + false positives.
- Recall :
    - The total number of true positives divided by the total number of target objects

	Image number	target bbox	target class	iou	predicted bbox	predicted class	confidence score	TP	FP	Cum TP	Cum FP
0	image 1	[198, 241, 215, 120]	Sheep	0.547604	[251.14699, 239.96185, 160.39119, 123.63736]	Sheep	0.990598	0	1	0	1
0	image 2	[302, 207, 138, 78]	Sheep	0.963164	[302.9632, 207.92993, 138.65057, 78.63748]	Sheep	0.983868	1	0	1	1
1	image 1	[389, 153, 100, 57]	Sheep	0.968898	[389.9533, 153.12675, 100.51538, 57.726486]	Sheep	0.979615	1	0	2	1
2	image 1	[428, 230, 97, 76]	Sheep	0.969188	[428.40094, 230.88051, 97.805115, 76.47783]	Sheep	0.963800	1	0	3	1
3	image 1	[198, 241, 215, 120]	Sheep	0.656665	[194.53662, 250.55005, 164.02713, 114.80841]	Sheep	0.947794	0	1	3	2
0	image 0	[228, 194, 70, 43]	Sheep	0.915433	[228.24442, 194.75839, 73.878784, 43.30397]	Sheep	0.942404	0	1	3	3
1	image 0	[448, 192, 89, 48]	Sheep	0.955063	[446.77637, 191.76418, 90.38379, 48.910934]	Sheep	0.921041	1	0	4	3
2	image 0	[295, 184, 96, 51]	Sheep	0.879498	[290.07745, 184.21776, 108.23135, 51.41365]	Sheep	0.901004	0	1	4	4
3	image 0	[337, 170, 31, 21]	Bird	0.892494	[337.4622, 170.9074, 31.930603, 21.825134]	Sheep	0.882748	0	1	4	5
1	image 2	[570, 238, 23, 22]	Bird	0.874526	[570.1732, 238.63919, 23.51825, 24.602142]	Bird	0.851861	0	1	4	6
2	image 2	[538, 259, 25, 26]	Bird	0.932697	[538.0018, 259.70435, 25.405945, 26.279755]	Bird	0.759862	0	1	4	7

- I calculated the cumulative true positives and false positives for each predicted bounding box and then calculated the precision and recall
- By changing the threshold of the IoU, new precisions and recalls list will be available.



- Then I calculated the average precision which is the area under the curve . I used the numpy.trapz trapezoidal rule.
- Then i get the mean of the average precisions for 0.5@0.05@0.95 = mAP = 0.3405605158730159

Problem 3:

- **What is word2vec?**

    - Algorithm to get the vector representation for words , so it presents words as vectors. These vectors captures the similarities between word -> similar words are close to each others
- **Where, how, and why is it used in this model?**
    - Was used to replace the sequence model (which was built from scratch).
    - Was used in the Embedding layer.
    - The embeddings were loaded and only the word vectors of the vocabulary of the descriptions was saved.
    - Also a fine tuned version of word2vec was used.
    - It was used as one of the steps to try and optimise the captioning model results.
- **Give a clear description and explanation of the loss function used for training this model.**
    - He used categorical cross entropy loss function which is used in multi-class classification , the model should choose one from some set of words (give probabilities)
    - It is a softmax activation and a cross entropy loss
    - The softmax activation function converts the model's output into a probability distribution where each class is assigned a probability from 0,1 and all classes's probabilities summing up to one .
    - Then the categorical cross entropy loss function to measure how close the y predicted from the true y.
    - https://medium.com/@vergotten/categorical-cross-entropy-unraveling-its-potentials-in-multi-class-classification-705129594a01
- **In evaluating the trained model, the author makes use of BLEU scores. What is a BLEU score, and how is it calculated? Do you think it is a sufficient measure for the quality of generated captions?**
    - Bilingual Evaluation Understudy is used to compare the similarities between the generated machine text and one or more ground truth texts.
    - Calculation of BLEU:
        - N-grams : continuous sequence of words
            - 1- gram => word by word
            - 2- gram => every pair of words
            - .. so on
        - Precision: the number of matched n-grams divided by the length of the generated text n-grams.
        - Global Average precision =>
        - $$Global\ Average\ Precision = exp(\sum_{n=1}^{N} w_n log p_n) = \prod_{n=1}^{N} {p_n}^{w_n}$$
        - Brevity penalty : applied when the generated text is shorter than the target text
        - $$Brevity\ Penalty = \begin{cases} 1, & \text{if c>r} \\ e^{(1-r/c)}, & \text{if c<=r} \end{cases}$$
        - r = number of words in the target text
        - c = number of words in the predicted text

    - Now to calculate BELU we just multiply the global average precision with the brevity penalty
    - Reference : https://www.scaler.com/topics/nlp/bleu-score-in-nlp/
- I think BELU is sensitive to the target text (2 sentences could have the same meaning with different words but they will get low score still)

Link of the notebook : omer_CV_Assignment_2.ipynb