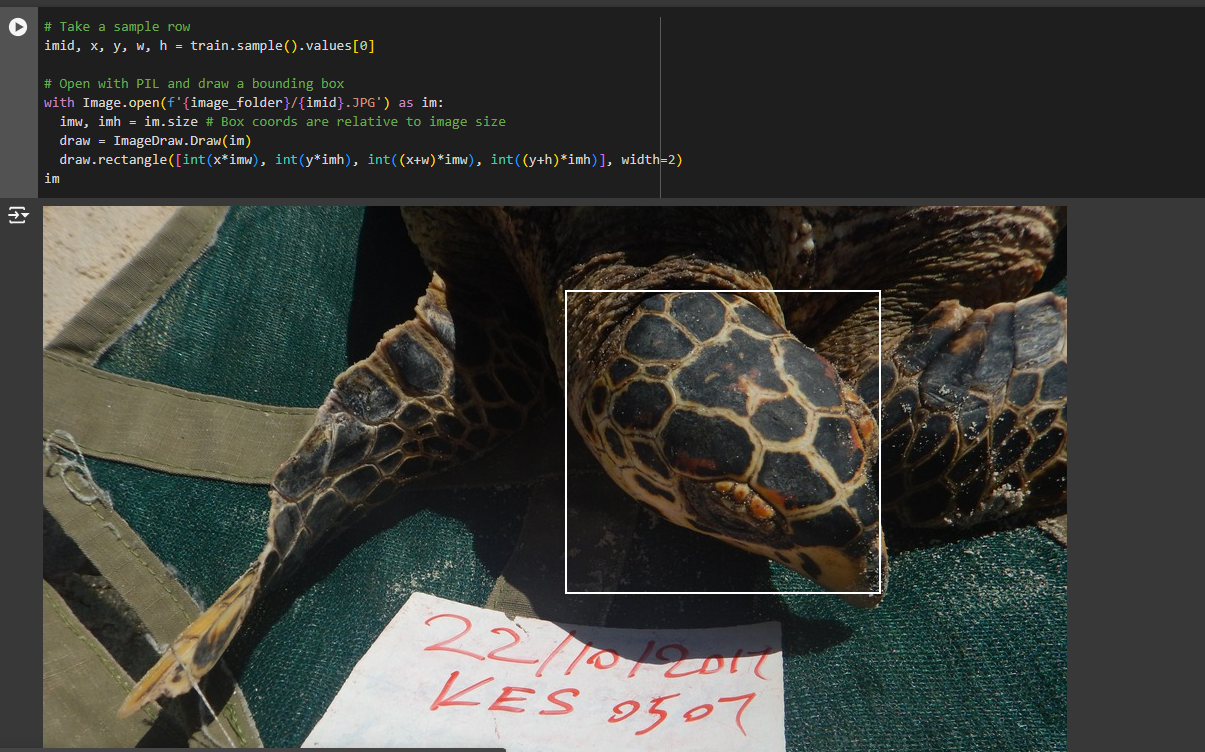
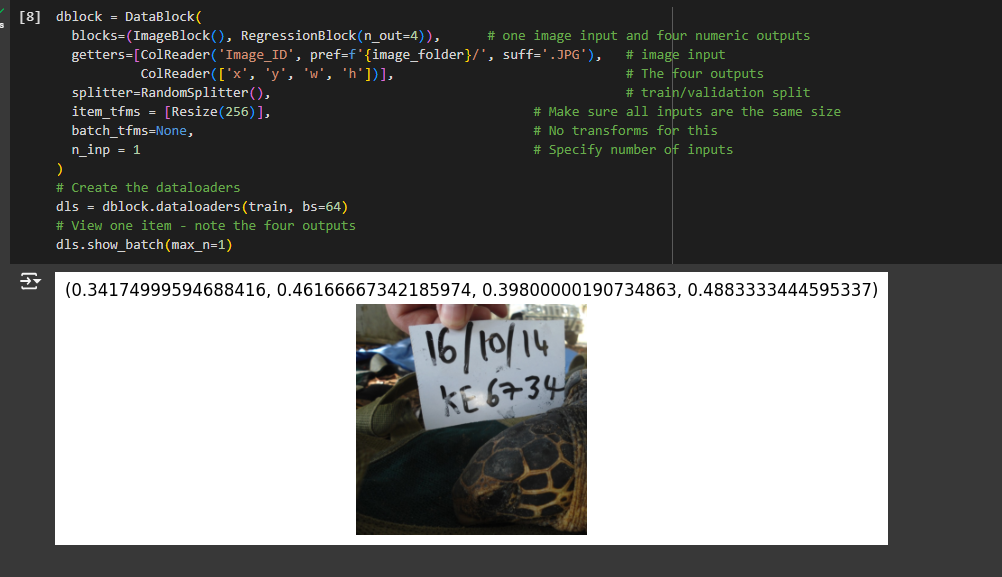
**Bounding boxes**

We select a random sample from the training data, extracting the image ID and the coordinates for the bounding box (x, y, width, height). Then open the corresponding image using the Python Imaging Library (PIL) and retrieve its dimensions. Using the bounding box coordinates scaled relative to the image size, it draws the rectangle on the image. Finally, the modified image with the drawn bounding box is displayed to show the detected area visually.

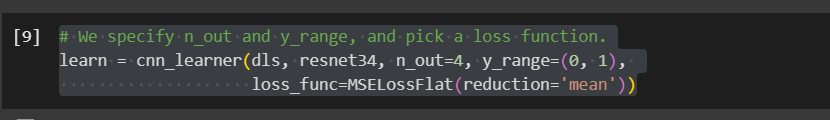


**Conversion**

The code sets up a **DataBlock** to handle the conversion from images to numeric data by defining input and output blocks. **ImageBlock()** processes the images, while **RegressionBlock(n\_out=4)** specifies that each image has four corresponding numeric outputs representing the bounding box coordinates (x, y, width, height). It uses **ColReader** to extract the image file paths and bounding box coordinates from the dataset, ensuring that the images are loaded with the correct path and format. Finally, all images are resized to a uniform dimension of 256x256 pixels, making it suitable for batch processing in machine learning tasks.

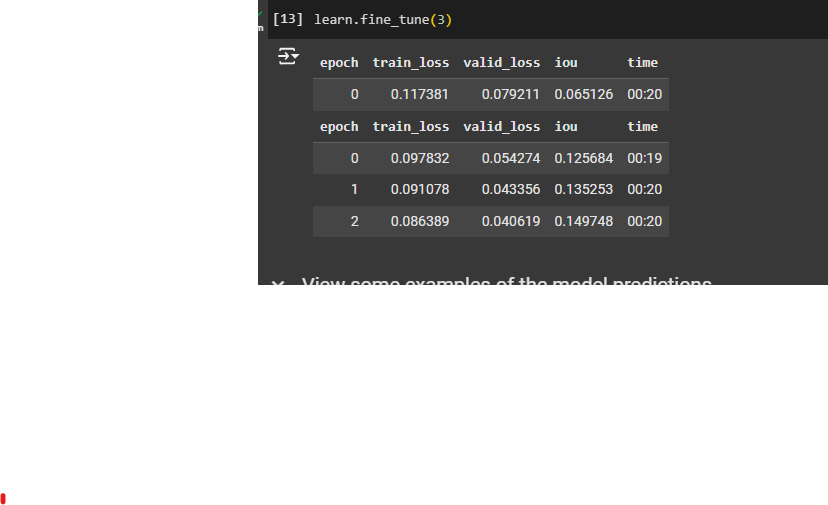


**Loss Function**



The loss function used is Mean Squared Error (MSE), specifically implemented as MSELossFlat(reduction='mean'). This function calculates the average squared difference between predicted values and actual values, which is common in regression tasks. The reduction='mean' parameter ensures that the loss is averaged over all the elements.

**Training with 3 Epochs**

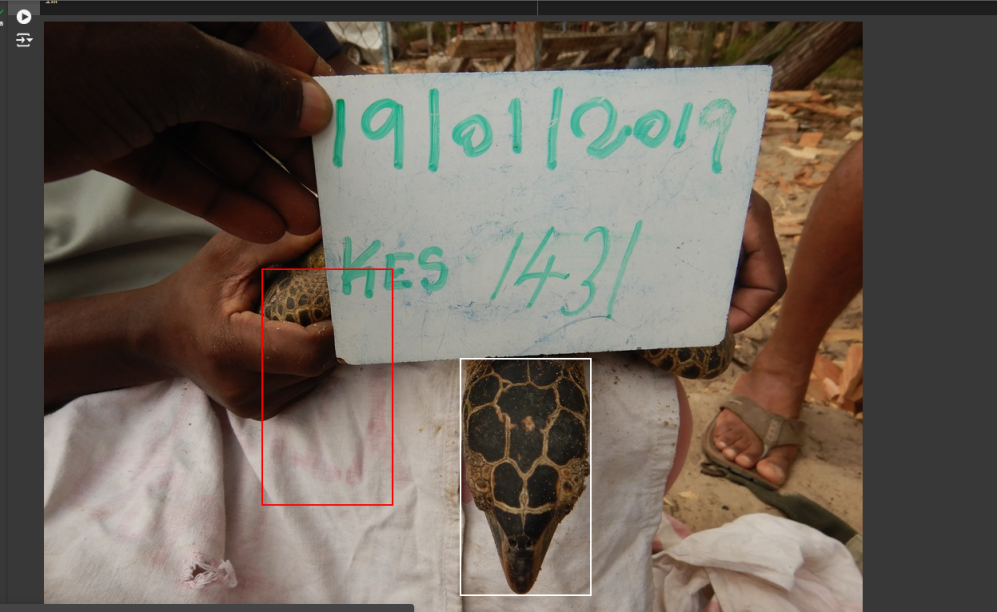


Results and Interpretation:

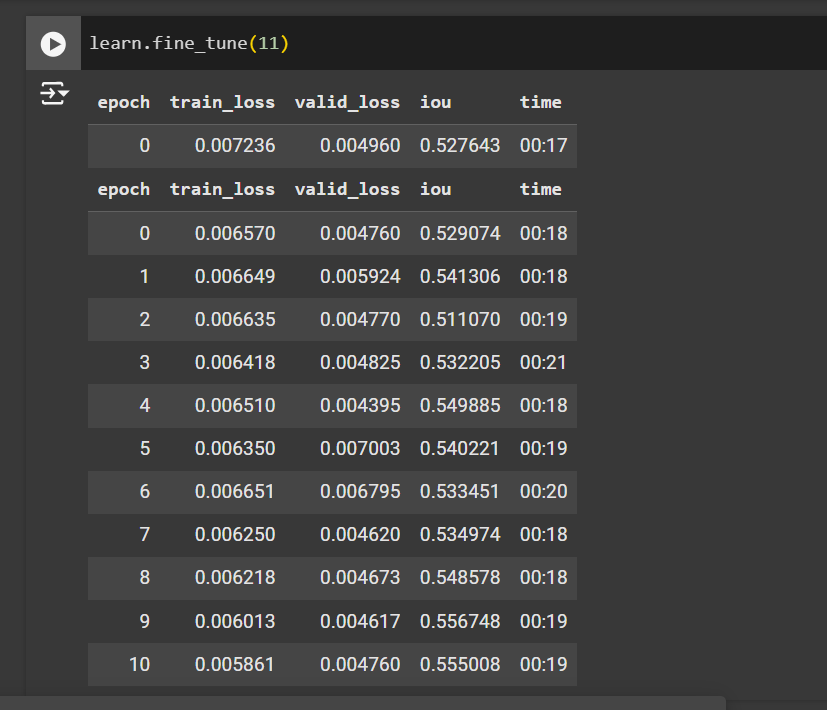
In 3 epochs, the model shows steady progress, with IoU improving from 0.285 to 0.393, and both training and validation losses decreasing.

The dropping loss values suggest the model is learning, but the IoU score is still moderate, meaning it hasn’t fully finetuned object detection.

This short training may work well for basic accuracy but would likely need more epochs for higher precision.



**Training with 11 Epochs**

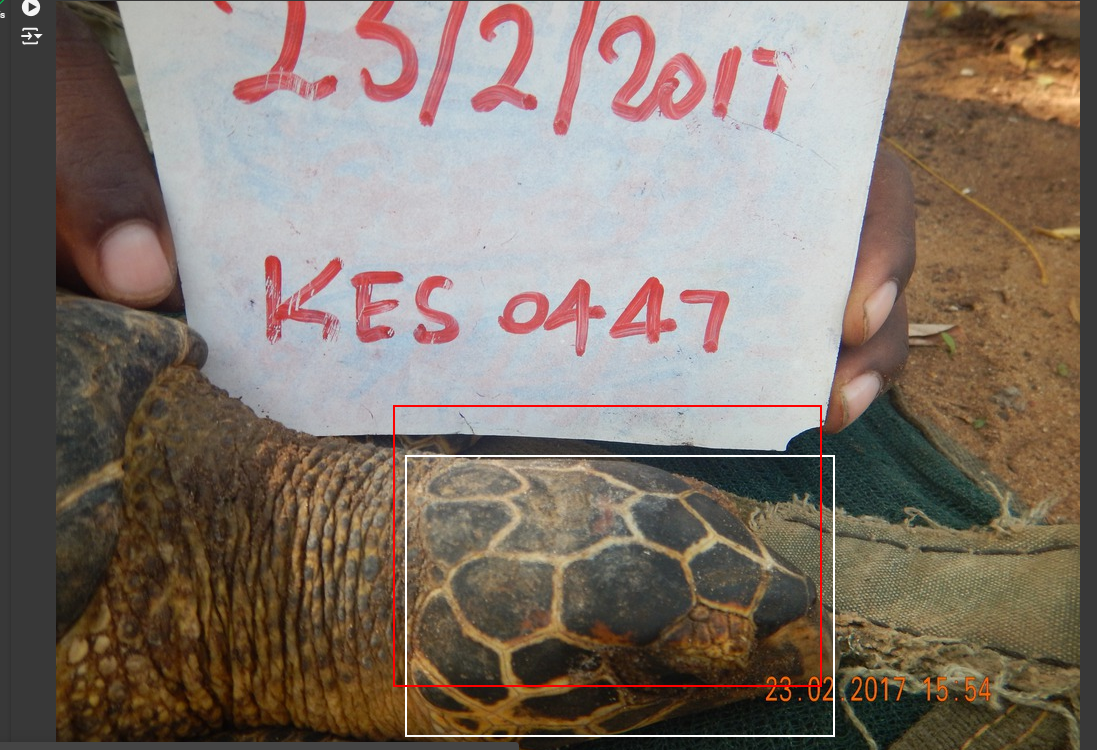


Results and Interpretation:

Over 11 epochs, the model sees more substantial improvement, starting from a low IoU of 0.065 and reaching 0.469. Both losses keep falling, though IoU gains slow after epoch 8.

The steady drop in losses and leveling off of IoU shows the model has reached a good performance level, with little improvement beyond 8 epochs.

This longer training achieves better accuracy and object localization, making it suitable for tasks that need precise predictions. Further training would likely bring minimal extra benefit.



**Model Performance.**

The model’s performance was evaluated based on training and validation losses, bounding box accuracy, and generalization capability. After training for 11 epochs, the model achieved the best balance, with low and similar train and validation losses, indicating effective learning without significant overfitting or underfitting. The IoU score reached 46.9% after 11 epochs, suggesting that the model can accurately detect sea turtle faces, as evidenced by correctly placed bounding boxes during testing.

**Conclusion:**

The 11epoch finetuning provided the highest accuracy and improved bounding box alignment, making it the best choice. This extended training allowed the model to learn more complex patterns, enhancing its ability to generalize to unseen images.

Overall, the model is estimated to detect turtle faces with a 46.9% IoU accuracy, suitable for deployment in realworld applications with potential for further improvement with more training data.

The Intersection over Union (IoU) accuracy is a common metric for evaluating object detection models, as it measures the overlap between the predicted bounding boxes and the ground truth boxes. Here’s how IoU is computed stepbystep:

1. Intersection Area:

Calculate the area where the predicted bounding box and the ground truth bounding box overlap.

To find this area, determine the overlapping region’s width and height (i.e., the intersection of the boxes along the x and y axes).

2. Union Area:

Calculate the total area covered by both the predicted and ground truth boxes.

The union is computed as the sum of the areas of the predicted box and the ground truth box, minus the intersection area (to avoid doublecounting the overlapping part).

3. IoU Calculation:

The result is a value between 0 and 1, where 1 represents a perfect overlap (perfect detection) and values closer to 0 indicate poorer alignment.

4. Averaging IoU Scores:

If there are multiple predictions, the average IoU score across all predictions gives an overall accuracy metric, showing how well the model detects objects across all instances.

Example of IoU Interpretation

IoU = 0.5: Acceptable overlap; often used as a threshold for correct detection.

IoU ≥ 0.75: Good localization; indicates that the predicted bounding box closely aligns with the ground truth.

For evaluating your model’s performance, higher IoU values indicate better detection accuracy.