Deep Learning Model Used

The model that was used was YOLO, which stands for You Only Look Once, is a real-time object detection system. Unlike traditional object detection methods that repurpose classifiers or localizers to perform detection, YOLO frames object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. This approach allows YOLO to achieve high accuracy and speed, making it suitable for real-time applications.

The YOLO model divides the input image into a grid of cells. Each cell is responsible for predicting a fixed number of bounding boxes and their corresponding confidence scores. These confidence scores reflect the probability that a bounding box contains an object and the accuracy of the bounding box itself. Additionally, each bounding box prediction includes class probabilities, indicating the likelihood of the object belonging to a particular class.

One of the key innovations of YOLO is its ability to predict multiple bounding boxes and class probabilities simultaneously, which significantly speeds up the detection process. The model uses a single neural network to process the entire image, making it more efficient compared to traditional methods that require multiple passes over the image. YOLO’s architecture typically consists of convolutional layers followed by fully connected layers, enabling it to capture both spatial and contextual information.

Overall, YOLO’s unique approach to object detection has made it a popular choice for various applications, including autonomous driving, surveillance, and robotics. Its balance of speed and accuracy continues to drive advancements in the field of computer vision.

Training a YOLO model involves several key steps. First, a large dataset of labeled images is required, where each image contains annotations for the objects present, including their bounding boxes and class labels, which are used to detect the animals. The model is then initialized with random weights and trained using a combination of convolutional and fully connected layers. During training, the input animal images are divided into a grid, and each grid cell predicts bounding boxes, confidence scores, and class probabilities.

The loss function used in YOLO training is a combination of localization loss to measure the accuracy of the predicted bounding boxes, confidence loss to measure the accuracy of the confidence scores, and classification loss to measure the accuracy of the predicted class probabilities. The model is trained using backpropagation and gradient descent, iteratively updating the weights to minimize the loss function. Data augmentation techniques, such as random cropping, scaling, and flipping, are often used to improve the model’s robustness and generalization.

Results

As a result of this project, we observed that YOLO offers several advantages, including speed, a unified architecture, high accuracy, and flexibility. However, the model also has some limitations, such as trade-offs between speed and accuracy, localization errors, and a complex training process.

YOLO’s speed can be attributed to its use of a single neural network to process the entire image in one go, unlike traditional object detection methods that rely on a pipeline of multiple stages (e.g., region proposal, feature extraction, classification). This unified architecture approach eliminates the need for multiple passes over the image, significantly accelerating the detection process. The model achieves high accuracy by leveraging this single neural network to process the entire image, capturing both spatial and contextual information. An impressive feature is its flexibility to detect multiple objects of different classes in a single image, making it versatile for various applications.

Based on this project, we observed that when YOLO makes predictions for certain videos, there are localization errors where it struggles to accurately localize small animals or animals that are too close to each other, resulting in lower precision metrics. Additionally, during training, there may be a need for high computational resources to achieve a perfect balance of speed and accuracy. Lastly, YOLO requires a large labeled dataset, which can be very labor-intensive when it comes to annotating bounding boxes for a large set of images.

Contributions

The team initially planned to annotate a set of images using a data annotation tool to create label files with bounding boxes. However, we quickly realized this approach would be very time-consuming and labor-intensive. To address this challenge, we used Roboflow to download multiple pre-labeled images and labels, which served as our custom training dataset. Dennis contributed by sourcing the initial set of animal images, formatting the input data according to YOLO’s requirements in a zip file, and drafting the Jupyter notebook code for data preprocessing, model training, and video prediction.