Object Detection using Faster R-CNN (Inference Option)

Link to the notebook:
https://colab.research.google.com/drive/1YgFSOlGfG4d4F7N0Pk5M1X5fW0ZX00Mm?authuser=0

Description

In this project, I trained and optimized an object detection model, specifically Faster R-CNN, on a custom dataset of aquarium images. I began by loading and pre-processing the dataset, which included applying data augmentations and transformations to improve the model's ability to generalize. Then I performed acceleration optimization on the model using ONNXRuntime and TorchScript, to see how they work and compared the performance of the 3 models.

Approach

1. Data Loading and Preprocessing

The dataset was loaded using a custom `AquariumDetection` class which inherits from 'datasets.VisionDataset'. This class is responsible for loading the images and their corresponding annotations (bounding boxes and labels) and applying the necessary transformations. Data augmentations such as resizing, horizontal flipping, vertical flipping, brightness contrast adjustment, and color jittering were used during training.

2. Model

I used a pre-trained Faster R-CNN model with a MobileNetV3 backbone, which was fine-tuned on our custom dataset. The model's box predictor was modified to predict the custom classes in our dataset.

3. Training

The model was trained for 15 epochs using Stochastic Gradient Descent (SGD) with a learning rate of 0.01, momentum of 0.9, and a weight decay of 1 x 10^(-4).

Optimization

1. TorchScript Optimization

The original model was converted to a TorchScript model using scripting. This statically-typed model was then evaluated in terms of inference speed and mAP.

2. ONNX Optimization

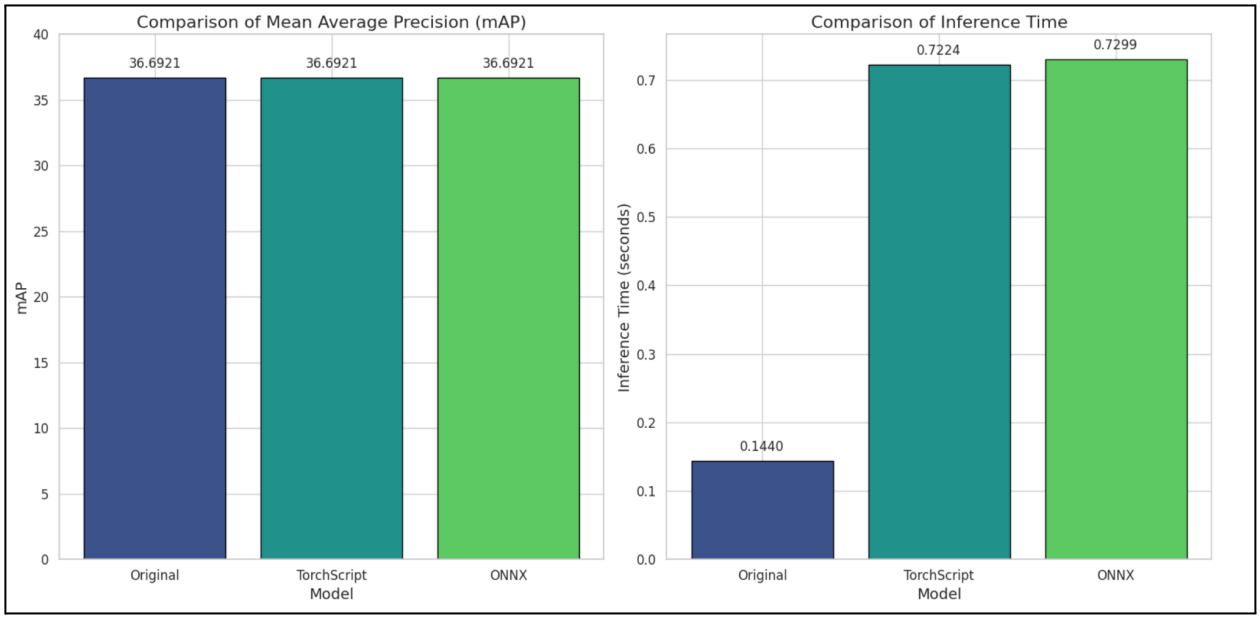
The model was also optimized using ONNX (Open Neural Network Exchange), providing a platform-independent representation of the model.

Evaluation Metrics

Mean Average Precision (mAP): The model's performance was quantitatively evaluated using the mAP metric across various IoU thresholds. This metric provides a comprehensive measure of the model's precision and recall capabilities.

Results

The performance of the original model, the TorchScript optimized model, and the ONNX optimized model were compared in terms of mAP and inference time.



Conclusion

While the mAP remained consistent across all three models, the increase in inference time for the ONNX and TorchScript optimized models was unexpected and warrants further investigation. Typically, optimization methods like ONNX and TorchScript are expected to improve or maintain the speed of inference, not reduce it. But given the very small size of the dataset(>1000 images), this behavior is expected, as these inference optimizations are going to significantly boost up the performance on a larger dataset, potentially consisting of video data.

References

1. https://pytorch.org/tutorials/recipes/torchscript_inference.html(TorchScript)
2. https://onnxruntime.ai/pytorch(ONNX)
3. https://www.kaggle.com/code/pdochannel/object-detection-fasterrcnn-tutorial
   (I have used some part of this code for the first part of the work, to train the model. I have made several changes to do things according to my work.)