AIP Assignment4

Badrinath Singhal (23302)

April 2024

JPEG Compression

We implemented peg compression by the isntruction given in the document. The size of the output file size 524288 bits and MSE we get are 2029.89. Whereas the compressino ratio is 1.

We see a big difference between custom JPEG compression and default jpeg compression. For custom jpeg compression we get MSE as 2029.86 with file size as 36361 bytes, whereas for default jpeg compression we get MSE as 15.165 with file size of 12242 bytes.

We made huffman table for lossless encoding for quantized coefficients to store in a encoded bit streams for each 8×8 DCT block.

Comparison of perceptual quality measures

Spearman rank order correlation coefficient is calculated with human evaluation for each of PSNR, SSIM and LPIPS metric. The results can be seen in table 1.

We found that spearman coefficient is lowest for SSIM metric whereas it was highest with LPIPs score. SSIM metric is better correlated with human visual evaluation whereas LPIPS and PSNR are not well correlated with human. But with PSNR and SSIM, higher the score better the results whereas with LPIPs lower score is better since it is a loss.

YOLO-v4

We trained yolo v7 using official implementation (https://github.com/WongKinYiu/yolov7) , Since the dataset was already given in the format which official implementation needs as the labels of objects for each image, we change the dataloader to include x,y,w,h. We created a new config file which contains the architecture of yolo network as well as other hyperparameters values. Network was trained using Nvidia-3090 GPU till we get decent class loss. Results can be seen in fig 2

The differences between YOLO v1 (taught in class) and YOLO v7 can be summarized in terms of architecture design, loss function, and computation complexity, along with how the changes proposed address any drawbacks in YOLO v1:

- **Architecture Design**: YOLO v1 consists of a single neural network that predicts bounding boxes and class probabilities directly from full images in a single evaluation. YOLO v7 typically incorporates improvements in the backbone architecture, such as using more advanced convolutional neural networks (CNNs) like ResNet or Darknet, eliminating fully connected

Metric	Spearman
PSNR	-0.66
SSIM	-0.91
LPIPS	0.951

Table 1: Spearman rank order correlation coefficient

Class	Images	Labels	Prec	Rec	mAP@0.5	mAP.5:.95
all	127	909	0.81	0.659	0.728	0.384
fish	127	459	0.852	0.675	0.775	0.387
jellyfish	127	155	0.76	0.837	0.845	0.436
penguin	127	104	0.72	0.669	0.667	0.279
puffin	127	74	0.678	0.581	0.594	0.265
shark	127	57	0.815	0.544	0.65	0.37
starfish	127	27	1	0.669	0.786	0.501
stingray	127	33	0.847	0.636	0.78	0.45

Table 2: Overall Performance of yolo v7

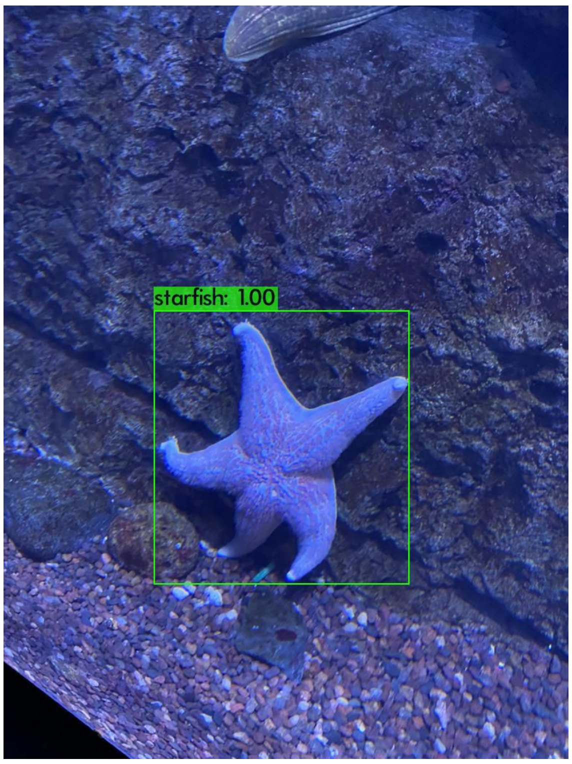
network. It may also include modifications to the detection head to improve accuracy and efficiency. Separate classification, regression and object confidence score network are made to resovle conflict.

- **Loss Function**: YOLO v1 uses a combination of localization loss (mean squared error) and confidence loss (binary cross-entropy) to optimize the detection performance. YOLO v7 introduce variations in the loss function to address specific issues or improve performance. For example, it may include focal loss to address class imbalance or incorporate other advanced loss functions to better handle overlapping bounding boxes. It also change the way we predict bbox (far from center, width as exponential function etc).
- **Computation Complexity**: YOLO v1 has a relatively high computation complexity, especially for large input images, due to its dense prediction strategy and single-pass evaluation. YOLO v7 aims to reduce computation complexity while maintaining or improving detection accuracy. This may involve optimizations such as network pruning, quantization, or efficient architecture design to accelerate inference without sacrificing performance. Removing dropout, one maxpool etc, VGG as backbone was used.

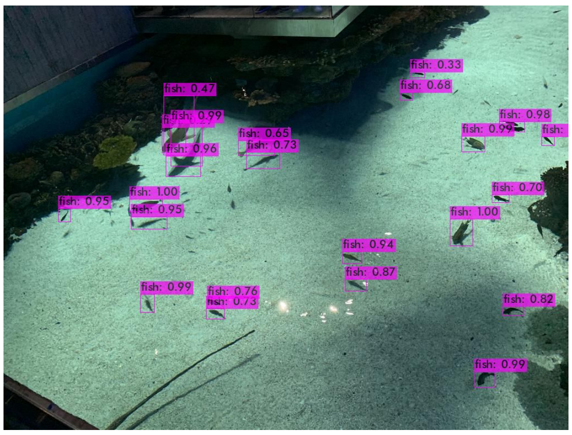
Justifying the changes:

- **Localization Accuracy**: YOLO v1 struggled with accurately localizing small objects and handling overlapping bounding boxes.
- **Computation Complexity**: YOLO v1's single-pass evaluation and dense prediction strategy led to high computation complexity, especially for large input images.
- **Architecture Design**: By incorporating more advanced backbone architectures like ResNet or Darknet, YOLO v7 can capture more abstract features and improve detection performance, especially for small objects.
- **Loss Function**: Introducing variations in the loss function, such as focal loss or other advanced loss functions, can help address issues like class imbalance and improve localization accuracy.
- **Computation Complexity**: Optimizations in computation complexity, such as network pruning, quantization, or efficient architecture design, can reduce the inference time and make YOLO v7 more suitable for real-time applications.

Average IOU for threshold is 0.3839, mAP for threshold 0.50 is 0.728



(a) STarfish example



(b) Fish detection

Figure 1: Examples of YOLO