Mobile Edge Object Recognition Application Measurements

Apostolos Galanopoulos

Abstract

This is a document complementary to the dataset describing the performance of the exemplary edge computing architecture for online object recognition on smartphones using a GPU-enabled edge server. Detailed information on the architecture itself is found in [1], while other related publications are [2,3]. Its purpose is to encourage the development of algorithms on optimal configuration of edge computing platforms using real data. To that end, this document briefly describes the system under study and discusses the obtained measurements that are provided in the supplementary file ”Edge-Dataset.csv”.

1 System Parameters and Measurements

The system is depicted in Fig.1 and is briefly described as follows. An Android application captures images though the mobile's camera, performs JPEG *encoding*, and transmits the compressed images[1] to a MEC server through a wireless 802.11ac Access Point (AP)[2]. A C/C++ routine at the server firstly decodes the JPEG files to obtain RGB images, and then downsamples them to match the input layer size of the Deep Neural Network (DNN) at the server's GPU. The integer RGB image values are converted to floats before processed with the state-of-the-art object recognition system YOLO [4], that accepts an $y \times y$ array of image pixels. Henceforth, the dimension y is referred to as the NN input layer size, or simply the *NN size*. The output is a set of: *(i)* bounding box coordinates, *(ii)* inferred classes, and *(iii)* confidence values for each recognized object. Those are transmitted back to the devices and overlaid on their screens. The main configurable parameters of this system are the image encoding rate x, which determines frame quality and file size, and the NN size y that affects the inference quality and delay.

1.1 YOLO and COCO image dataset

The initial attempt to analyze the system and its performance is thoroughly described in [1]. YOLO accepts an image size that is a multiple of 32 and uses a number of potential object locations of different sizes to output a probability of each class in the training set to appear in each one. However, determining the actual recognition performance of YOLO we need ground truth information, i.e. knowing the actual class and bounding box coordinates of the test image and compare it with the output of YOLO.

[1]Encoding an image at a certain rate, e.g. 50% achieves 2 things. First, the image data is converted to the JPEG format, and second the resulted file is compressed to 50% of the original file size, improving transmission delay but hampering image quality.

[2]The AP is the ASUS RT-AC86U router, and the server a 3.7 GHz Core i7, 32 GB RAM PC, with a GeForce RTX 2080 Ti GPU.

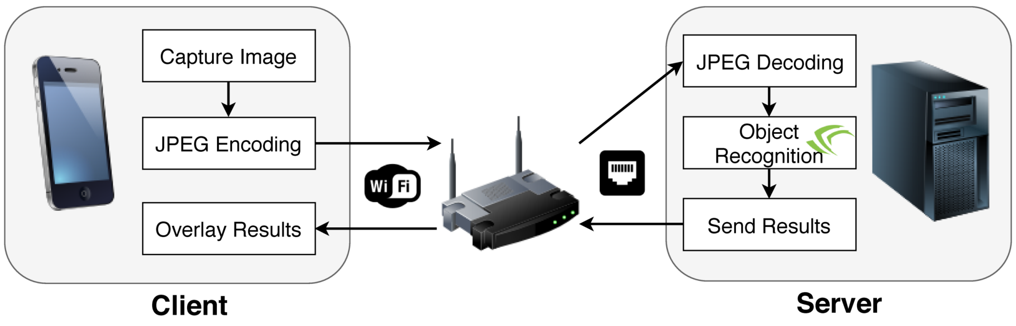


Figure 1: Architecture of the edge object recognition system.

We used the extensive COCO dataset [5] which covers a wide range of images and objects, and includes ground truth for each image (object locations and labels within each image). For quantifying performance, we used the Average Precision (AP) and Average Recall (AR) metrics. AP is the ratio $T_p/(T_p + F_p)$ while AR is the ratio $T_p/(T_p + F_n)$, with T_p being the true positive detections, F_p the false positive and F_n the false negative detections. The results are averaged over all objects classes and can be easily obtained with the help of the *"COCO"* and *"COCOeval"* python modules.

To use the COCO images we connected the phone to a server via a USB cable and a Python script on the server sends commands to the phone using the Android Debug Bridge (adb). The server initiates the client application through adb and configures the system parameters for the experiment (e.g., the JPEG compression level). Then it iterates over 5000 images from the COCO validation set, sending them one-by-one to the phone through cable. The phone transmits each image to the server through the wireless interface, as if they were images captured by its camera, receives the server response over WiFi and passes this back over the USB cable for logging.

1.2 Online setup and measurements

The main problem with the AP and AR is that they require offline information such as the number, location and classes of objects in each transmitted image. An online system however, is targeted at making optimal configuration decisions based only on information that can be extracted by each image as it passes through the YOLO DNN. As stated earlier, YOLO outputs a set of bounding box coordinates, inference confidence (probability), and class for each recognized object in an image.

Towards obtaining a rough but online measure of recognition performance, we consider the sum of inference confidence for all recognized objects as a recognition performance metric. This implies that as the number of recognized objects and their respective inference probability increase, so does the confidence sum, which we call Cumulative Confidence (CC). In [2] we explain why this is a good approach, and we visualize the similarity with offline metrics AP and AR in Sec. 2.

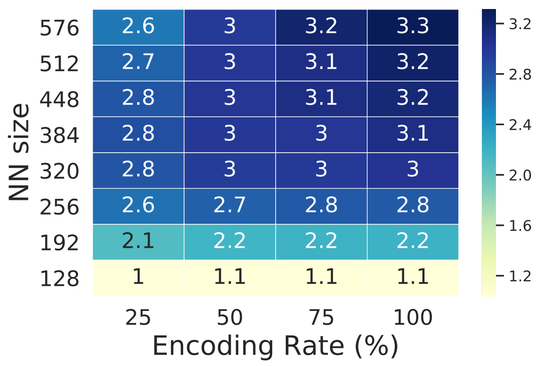
On top of object recognition performance, we are interested in the system's latency response for the different values of NN size and encoding rate. To that end, the Edge-Dataset includes 8 possible values for the NN size (128,192,256,320,384,448,512,576) and 4 for the encoding rate (25,50,75,100). For each NN size/encoding rate combination we test the response of 1000 images from COCO[3] and measure the latency response in each part of the process, as well as the confidence values that are output from the DNN. In detail, the first 2 columns of each record, are the "NN size" and "Encoding rate" respectively. They are followed by columns "Encoding", "Network", "Decoding","Rotating",

[3]Hence the Edge-Dataset consists of 32000 records in total.

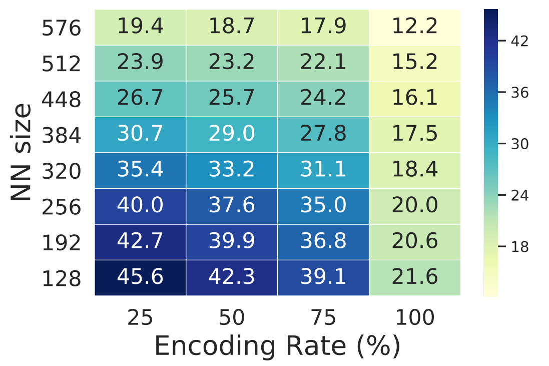
”yolo”, and ”Server”, which track the amount of time spent in milliseconds for each of these tasks, where ”Server” is the total time spent on server processing including YOLO processing on the GPU. Column ”Frame Rate” is simply the frame rate achieved for the image’s end-to-end latency calculated by inverting the latter. The last 2 columns measure recognition performance and are the ”Confidence” and ”Cumulative Confidence”. Confidence is an array of values $\in [\,]0, 1]$, denoting the inference confidence for each identified object in the image. If no objects are recognized the array is empty. Cumulative Confidence is simply the sum of the Confidence array.

2 Result Visualization

In file *process_data.py* you can find the code to visualize the dataset as follows. Figures 2a-2b display a heatmap of the average (across 1000 images) Cumulative Confidence and frame rate respectively, for each NN size/encoding rate combination. Notice how areas with high Cumulative confidence have low frame rate, and vice versa, indicating the trade-off between the 2 metrics.

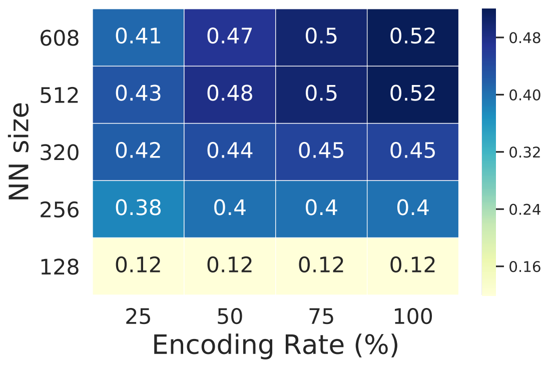


(a) Cumulative Confidence

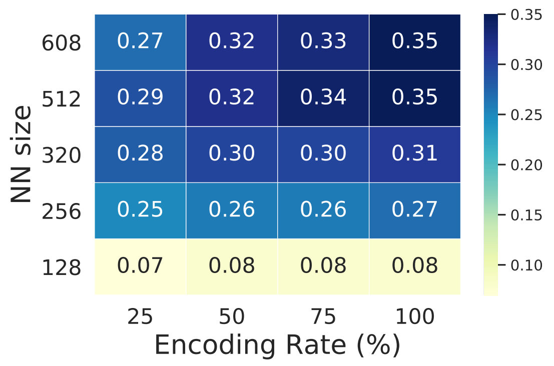


(b) Frame Rate

Figure 2: (a) Cumulative Confidence and (b) frame rate for various NN sizes and encoding rates; results are averaged across $32K$ images of COCO dataset.



(a) Average Precision



(b) Average Recall

Figure 3: (a) Average Precision and (b) Average Recall for various NN sizes and encoding rates; results are averaged across all cross-validation images of COCO dataset.

To highlight the suitability of Cumulative Confidence as an online alternative to AP/AR, Figures 3a-3b display the respective heatmaps (with slightly different values for the NN size). Observe the qualitative similarity between those 2 figures and Figure 2a. Note also, the strange behavior at 25% encoding rate, where the Cumulative Confidence/AP increases with the NN size until the value of 448/512 respectively, and then decreases. These similarities prove that the Cumulative Confidence is a suitable metric to replace AP/AR when the latter is unavailable due to incomplete (unlabeled) data or an online setup.

References

[1] A. Galanopoulos, V. Valls, G. Iosifidis, and D. J. Leith, "Measurement-driven Analysis of an Edge-Assisted Object Recognition System," in *Proc. of IEEE ICC*, 2020.

[2] A. Galanopoulos, J. A. Ayala-Romero, G. Iosifidis, and D. J. Leith, "Bayesian Online Learning for MEC Object Recognition Systems," in *Proc. of IEEE GLOBECOM*, 2020.

[3] A. Galanopoulos, J. A. Ayala-Romero, D. J. Leith, and G. Iosifidis, "AutoML for Video Analytics with Edge Computing," in *Proc. of IEEE INFOCOM*, 2021.

[4] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[5] T. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *arXiv*, vol. abs/1405.0312, 2014.