Face Detection at 15,000 FPS: Real-Time Inference on GPU and CPU

Matheus Alves Diniz[1], David Menotti[2], William Robson Schwartz[1]
[1]Smart Surveillance Interest Group, Smart Sense Laboratory
Department of Computer Science, Universidade Federal de Minas Gerais
[2]Department of Computer Science, Universidade Federal do Paraná
matheusad@dcc.ufmg.br, menotti@inf.ufpr.br, william@dcc.ufmg.br

***Abstract*—Object detection is a key task in computer vision since it is the first step in the pipeline of many applications such as person re-identification, vehicle identification, and face verification. Recently, the best performing object detectors have been achieved with deep learning and one common characteristic among them is that they are a very slow on ordinary hardware. Reported real time object detectors are usually measured with high-end GPUs, which is inappropriate for scenarios where energy efficiency and low costs are required. We were able to train a very light face detection architecture by greatly reducing the number of parameters and input size of a convolutional network. Our model is capable of performing inference in real time on a hardware as simple as a Raspberry Pi. Furthermore, when evaluated on a GPU, we were able to achieve up to 15,000 frames per second.**

I. Introduction

When Krizhevsky [1] won the ILSVRC-2012 [2] image classification challenge, the rise of deep learning (DL) in computer vision followed. However, the strong performance of deep learning comes with an expensive computational cost. Inference time is usually reported with high-end GPUs but many models fail to run at real-time.

The dependency on expensive GPUs is also a major hindrance for the financial viability of computer vision in real applications. Furthermore, embedded and mobile systems are usually equipped with energy efficient hardware, which usually lacks processing power to run these models in real-time.

In this paper, we investigate object detection with convolutional neural networks. Our main focus is to develop applications that are able to run in real-time on a modest hardware. We believe that by bringing down the cost of the models, deep learning could be applied to broader spectrum of problems. In addition, our techniques can also be used in the classroom: our model follows the same principles and develops the same insights as larger models, but it can be trained in an ordinary computer, which might be an important tool for teaching deep learning to students.

We chose object detection as our focus due to its applicability in many different scenarios. For instance, detection of cars, license plate, faces and pedestrians have direct applicability on automatic license plate recognition, face verification and person re-identification. Many tasks are also directly related with detection, and as far as we know, no effort has been previously made to develop and evaluate possible scenarios for real time CPU deep learning models for object detection.

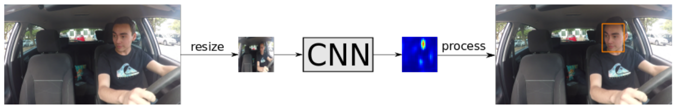

Fig. 1. Our approach resizes the original image down to a miniature, where the convolutions can be performed much faster. While small faces could be lost on the interpolation process, this may not be a problem when the application only cares about faces above some size. For instance, faces of pedestrians outside the car are too small, but they may not be of interest.

Our approach focuses on small inputs images. Recent object detectors [3] have been able to detect faces which are only a few pixels tall. We argue that face detection could be performed on smaller inputs as long as the faces on the original image have enough resolution to not disappear in the smaller image. Fig. 1 illustrates the steps of our approach.

To validate our approach, we collected a set of videos inside a car cabin. We were able to achieve very strong performance in this set of videos, and furthermore our method was able to run in real time in a common CPU. Our proposed model uses less than 10k parameters, i.e., orders of magnitude smaller than conventional models. As a result of such model, our approach is able to detect more than 15,000 frames per second (FPS) using a GTX 1080TI, more than 250 FPS using an i3 core CPU processor and 49.61 FPS using a Raspberry Pi.

II. Related Works

Object detection is a crucial step in many computer vision applications as well as an important application by itself. Deep learning approaches to this task can be divided into two categories: two stage detectors and one stage detectors.

Two stages detectors rely on a region proposal network step to generate candidate regions for objects in the image. Each of these candidate regions is then classified as one of the objects classes, or background. These two steps can be done separately [4], but using a single model decreases detection time [5] and greatly increases accuracy [6].

One stage detectors, also known as single shot detectors [7], skip the region proposal step and directly classify the presence of objects in a fixed set of candidate regions [8]. This leads to faster inference time at the cost of small drops in accuracy, though this accuracy gap seems is becoming ever smaller [9]

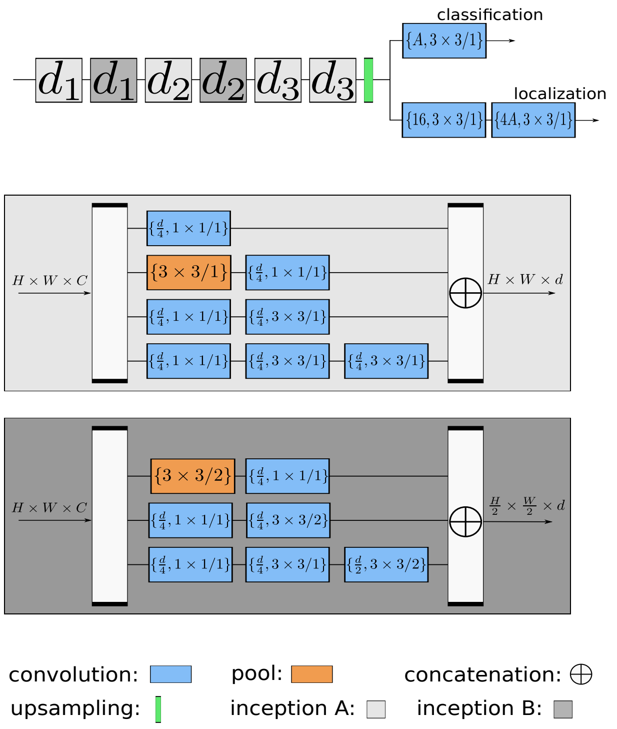

Fig. 2. Our detection network architecture (on top). Inception-A, and Inception-B blocks are detailed in the middle, inside the light-gray and dark-gray boxes respectively. The depth of each block (d_1, d_2, d_3) is composed by the concatenation of the convolutional towers inside the inception blocks. The variable A indicates the number of anchors used at each point in the grid. Convolutional layers are described by {filters, kernel size/stride}, and pooling layers, by {kernel size/stride}. Each convolution is followed by batch normalization and ReLU, but they are omitted for better visualization.

In [10], object detection parameters, such as number of anchors and input size, are studied. Different backbones are also evaluated as feature extractors for three distinct object detectors: Faster R-CNN, R-FCN, and SSD. While the focus of the study is to analyse speed and accuracy trade-offs, even the fastest model evaluated would be still too heavy for real-time computation on CPU.

Howard et. al [11] studied efficient deep learning models in order to get high performance out of light weight models. They identify two hyper-parameters to control model size and accuracy, namely α, which controls the width of the networks and ρ, which controls the input resolution. They also propose depthwise separable convolution as an alternative to the standard convolution in order to decrease the number of float operations per convolutional layer.

While there are many works that investigate efficient deep learning architectures, they still are not able to achieve real time CPU because the focus of their analysis is still on robust models and datasets. In this work, we evaluate a more restrictive domain which allows us to greatly reduce our architecture size, and thus achieve real-time CPU detection.

III. Proposed Approach

We developed a deep learning architecture capable of performing detection in real time even when only a simple CPU is available. We focus our evaluation on a specific domain: face detection inside a car cabin. In this scenario, the faces are expected to have large sizes and our model does not need to be robust to large scale variances, which allows us to greatly reduce the model size and inference time.

To minimize inference time, it is necessary to reduce the number of floating operations performed by the network. This can be achieved in at least three different ways: (i) reducing the number of layers in the model; (ii) reducing the number of channels in each layer, also known as the model width; or (iii) reducing the input size of each layers.

Reducing the input size of the layers is equivalent to reducing the input size of the model. It has been shown that exceptionally small objects can be detected with deep learning [3]. Therefore, if we investigate a domain where only large objects are present, we can downsample the domain input and expect to maintain the performance of small objects. For instance, if a robust model is able to detect faces with height of 6 pixels and we are interested in detecting faces in an environment where faces are no smaller than 60 pixels, we can expect that the face will still be detected if we downsample the image by a factor of 10. One possible explanation for this is that the faces with large resolution presents too much redundant information which is not necessary for detection. In addition, smaller inputs also allows the network to use fewer filters on the convolutional layers since the image information becomes more concentrated in fewer pixels. Thus, reducing the input size also favors the reduction of the model width.

Our model is summarized in Fig 2. Since our goal is to speed up the inference time, we follow the one-stage detectors strategy and we adopt a fixed set of candidate regions. The inception [12] backbone also helps with this goal because it grants a deeper architecture for less computational budget. Residual connections [13, 14] between the blocks were evaluated but not used in the final model because they provided a significant overhead in inference time while offering no major accuracy improvement. Since the input resolution is very small, we also upsample the last layer of the backbone to increase the size of the feature map, and consequently the number of candidate regions available for detection.

The detection sub-model classifies each candidate region as either a face or a background. Therefore, to train this sub-module, we generate a classification grid for each one of A the anchors. We assign 1 to positions where the face has intersection over union (IOU) greater than 0.45. Positions where the intersection is below 0.3 are assigned 0. Finally, anything between 0.3 and 0.45 is set to *ignore* and does not contribute to the loss function. We use focal loss [9] for this task, which is a modification on the more commonly used cross-entropy that down-weights the impact of easy classifications (e.g. easily classified background regions) that are common in the object detection scenario.

The localization sub-model adjusts each of the anchors to better match the actual bounding boxes. Thus, for each anchor grid in the detection sub-model, there are 4 grids in the localization sub-model, each one regressing one of the

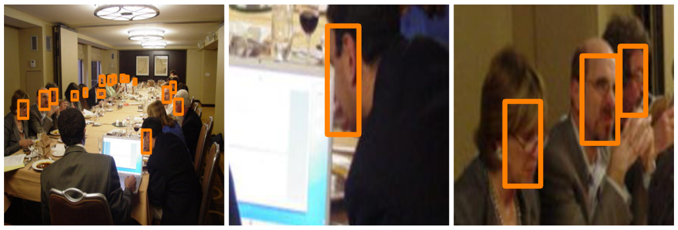

Fig. 3. On the left, the original image of WIDER-FACE dataset. On the center and on the right, modified crops in which the face size matches our targets.

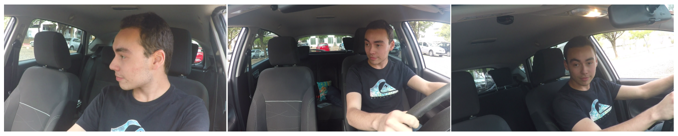

Fig. 4. Our test dataset consists of videos filmed with three *GoPros* attached to the windshield on the driver (left), center (middle), and passenger side (right).

4 coordinates of the bounding box. For this sub-model, we employ the smooth-L1 as our loss function.

IV. Experimental Results

Unless stated otherwise, all reported experiments were performed in a machine with processor *Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz*, 8 GB of memory, and no GPUs.

Dataset. Since we are only interested in faces of a specific size, we need training samples that match these sizes. Instead of collecting samples on a similar scenario, we apply modifications to an existing dataset so that it fits our constraints. The main benefit of this approach is that it avoids costly annotations and data acquisition. It also helps to make the trained model more robust to different scenarios since the training and the testing data are not captured equally.

We use WIDER-FACE [15] as the underlying dataset in which we make our transformations. Our approach takes an input image from the original dataset and crops it so that the resulting image has a face with dimensions matching to the ones in our domain. For the car cabin that would be crops in which the face height is between 25% and 50% of the total image height, but the method could be adapted to a broader or a more restrictive spectrum. Fig. 3 shows the original image and two samples generated by this method.

During training, each image on the training set of WIDER-FACE is modified and fed to the network, though the sample may differ at each execution, since there may be more than one face per image. On the other hand, the test set of WIDER-FACE cannot be adapted to our domain because the ground truth is not provided. Thus, we collected a dataset of car cabin videos to evaluate our model.

A total of 10 individuals were filmed on the driver seat of the car from three different angles, as shown in Fig. 4. Each video has, on average, 30 seconds, filmed at a rate of 30 frames per second. At the video capture, the subjects were instructed to look at each of the mirrors in the car in order to introduce pose variation in the video.

Evaluation of Our Approach. Overall, we are able to achieve a network configuration with less than 10,000 parameters, more than 85% of these belonging to the localization and detection sub-models, which correspond to only 3 out of the 43 convolutional layers. For this setup, we empirically set d_1, d_2, d_3 to 4, 8 and 12, respectively. Since WIDER-FACE contains a large variation of poses, we have to use 9 anchors in order to be able to match all of them. Less anchors could be used if the domain was even more restrictive, as if it contained only frontal faces, or an even narrower height range.

For students, one of the key advantages of our model is that it can be trained from scratch in just a few hours on the *i3* processor. Learning rate was set initially to 0.0005, batch size to 256, and ADAM was used as our optimizer. The learning rate was also set to decrease by a factor of 10 whenever the loss did not decrease for 2 consecutive epochs. With this configuration, we were able to train our model in less than 3 hours for 70 epochs.

Figure 5 shows the precision recall curve for our model when the input size is set to 48×48 pixels. In our first evaluation, we select 50 frames from each video and perform detection varying the classification threshold over different IOU metrics. The classification threshold controls the value of the output of the detection sub-model that separates background from faces.

At 0.5 IOU, which is the traditional value in many datasets, we are able to achieve recall and precision above 80%. Considering that our model was developed to be very light, and thus run at every single frame, it would be very unlikely to miss the face in a video segment for many consecutive frames. Furthermore, the 0.3 and 0.4 IOU curves show that many of the false positives and false negatives in our model are due to poor localization instead of miss-classification, i.e. the face is detected but the bounding-box is not perfectly fitted. For the rest of the evaluation, unless specified otherwise, our measures consider only IOU above 0.5 as a positive match.

Table I compares our model with different input sizes against Viola & Jones [16], which employs a cascade of classifiers to detect faces in an image pyramid. Our choice for this method is based on the fact that, though outdated, this method is known as a very fast detector. We used the implementation available at OpenCV to run our evaluations. While Viola & Jones is able to achieve real-time in our CPU, it shows poor performance, specially when the face is not fully frontal. Combining a second cascade classifier for profile faces improves the detection rate but it is still not able to achieve comparable results both in accuracy and inference time. While reducing the input size for Viola & Jones does reduce inference time, its handcrafted features are not able to distinguish between background and faces at a resolution as small as our model does.

Our model also seems to lose performance at the 80×80 pixels input resolution. One possible explanation is that this is due to the fact that we kept the network width and depth constant throughout the experiments. At this resolution, it is possible that the small backbone lacks capacity to learn all

TABLE I
AVERAGE PRECISION (AP) AND RECALL (AR), AND FRAME RATE FOR EACH MODEL. BATCH SIZE EQUALS 1, UNLESS SPECIFIED.

Method (resolution)	AP	AR	Frames per second CPU I3 6006U	Raspberry Pi	GTX 1080TI	GTX 1080TI$_{batch=256}$
Ours (16×16 pixels)	0.65	0.53	**277.76**	**49.61**	**130.21**	**15,442.34**
Ours (32×32 pixels)	0.77	0.75	182.65	38.31	117.15	7,811.34
Ours (48×48 pixels)	0.85	0.83	120.36	25.91	122.64	4,064.50
Ours (64×64 pixels)	**0.89**	**0.84**	82.37	16.86	99.64	2,866.93
Ours (80×80 pixels)	0.84	0.80	60.07	11.61	105.83	1,092.50
Viola & Jones frontal (480×300)	0.83	0.25	50.61	9.69	-	-
Viola & Jones frontal+profile (480×300)	0.81	0.37	22.98	4.36	-	-
Viola & Jones frontal+profile (64×64)	0.60	0.00	258.60	45.12	-	-

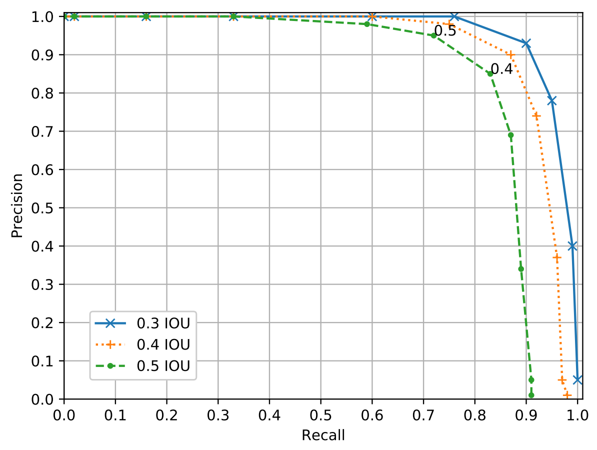

Fig. 5. Precision recall curve for our model with 48×48 input size. Using the conventional 0.5 intersection over union (IOU) metric, we were able to achieve a good recall and precision in our test set. The curves with lower thresholds, show that many of the misses are due to poor localization, rather than miss-classification. On the top of the green curve, we show the classification thresholds 0.5 and 0.4, which are ideal for our model.

necessary features to detect faces in the larger and sparser grid. Thus, it is likely that increasing the model width, i.e. the values for d_1, d_2, d_3 could improve the performance of this model, but at the cost of a drop in FPS.

We also evaluate the performance of our proposed detector on a Raspberry Pi 3. This test shows that it is feasible to use our model on embedded systems and mobile hardware. The two smaller models were able to be above real time, while the 48×48 model was nearly real time.

When evaluated with a GPU the model is sometimes slower than on CPU. This can be explained by the overhead necessary to transfer the image to the GPU memory which is not negligible at this frame rate. This hypothesis can be corroborated by the evaluation on larger batches. At a batch size of 256, we are able to achieve extremely fast detection, as high as 15,000 FPS with a high end GPU.

V. CONCLUSIONS AND FUTURE WORKS

In this work, we were able to train an efficient model for face detection. Though our domain is restrictive (i.e., smaller faces are not detected) when compared to other datasets, it is not unrealistic. This same principle could be applied to other computer vision tasks that could benefit from specialized models and achieve faster inference speeds. In this sense, vehicle detection is a promising target since cars have reasonable size in comparison to the whole scene, such as a single lane roads.

We also showed that our approach may aid interested students in the development of their own models. Excessive training times is one of the biggest obstacles to beginners in deep learning. With this approach, it is feasible for students to train their models overnight on their own laptops and achieve similar results, or even tune our model even further.

ACKNOWLEDGMENTS

The authors would like to thank the Brazilian National Research Council – CNPq (Grants #311053/2016-5, #428333/2016-8 and #313423/2017-2), the Minas Gerais Research Foundation – FAPEMIG (Grants APQ-00567-14 and PPM-00540-17), the Coordination for the Improvement of Higher Education Personnel – CAPES (DeepEyes Project), Maxtrack Industrial LTDA and Empresa Brasileira de Pesquisa e Inovação Industrial – EMBRAPII.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.
[2] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, 2015.
[3] P. Hu and D. Ramanan, "Finding tiny faces," in *CVPR*, 2017.
[4] R. Girshick *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.
[5] R. B. Girshick, "Fast R-CNN," *CoRR*, 2015.
[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *TPAMI*, 2017.
[7] W. Liu *et al.*, "SSD: single shot multibox detector," *CoRR*, 2015.
[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.
[9] T. Lin *et al.*, "Focal loss for dense object detection," *CoRR*, 2017.
[10] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *CVPR*, 2017.
[11] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, 2017.
[12] C. Szegedy *et al.*, "Going deeper with convolutions," in *CVPR*, 2015.
[13] K. He *et al.*, "Deep residual learning for image recognition," in *CVPR*, 2016.
[14] C. Szegedy *et al.*, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, 2016.
[15] S. Yang *et al.*, "Wider face: A face detection benchmark," in *CVPR*, 2016.
[16] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.