Towards Real-time Video Content Detection in Resource Constrained Devices

Jhonatan Geremias
Graduate Program in Computer Science - PPGIa
Pontifical Catholic University of Parana
Curitiba, Brazil
jhonatan.geremias@pucpr.br

Altair O. Santin
Graduate Program in Computer Science - PPGIa
Pontifical Catholic University of Parana
Curitiba, Brazil
santin@ppgia.pucpr.br

Eduardo K. Viegas
Graduate Program in Computer Science - PPGIa
Pontifical Catholic University of Parana
Curitiba, Brazil
eduardo.viegas@ppgia.pucpr.br

Alceu S. Britto Jr.
Graduate Program in Computer Science - PPGIa
Pontifical Catholic University of Parana
Curitiba, Brazil
alceu@ppgia.pucpr.br

***Abstract*—Convolutional neural networks have been successfully applied for video content detection in the last years. However, such cognitive models usually demand the availability of several gigabytes of memory and present a low detection throughput, as a result, they are not feasible for resource-constrained devices, especially for real-time applications like video streaming. In this paper, we address real-time video content detection in resource-constrained devices in a threefold manner. First, we improve detection throughput by means of a frame sampling technique. Then, we propose a new evaluation measure towards proper deployment of convolutional neural networks in resource-constrained devices. Finally, we address the accuracy degradation caused by the porting of the convolutional neural network, applying a lightweight classification verification technique. The evaluation results, through a real-time demanding application, show that the proposed approach can detect up to 301 frames/sec, demanding only 9 megabytes of memory while reaching up to 89.3% of accuracy. Besides, we can increase the detection throughput by up to 10 times, with no effects on accuracy, and further increase accuracy without effects on processing demands.**

***Index Terms*—Neural Networks, Video Content Detection, Resource Constrained Devices**

I. Introduction

Currently, there are about 3.2 billion of smartphones [1], with the expectation to reach the 3.8 billion mark in 2021 [2]. Smartphones are considered processing-constrained devices, therefore the processing tasks that were typically designed for energy-powered devices must be ported to resource-constrained ones, for instance, real-time pornography content detection in videos [3].

Video content detection in real-time is a costly processing task, which, in general, demands a significant portion of the device processing capacity [4]. Over the last years, video content detection has been achieved by machine learning means, more specifically through *Convolutional Neural Networks* (CNN) [5]. CNN has been widely used in image processing tasks due to its high reported accuracy rates [6]. However, they were not originally designed for processing-constrained devices, in contrast, they are often memory and processing demanding [7]. On the other hand, for video processing tasks, a CNN architecture must evaluate each frame individually. As a result, to perform online video content detection for instance, in general, it is needed to process up to 23 frames/sec. Therefore, perform real-time video processing (e.g. in video streaming applications) through typical CNN architectures at processing-constrained devices becomes a challenging task. In such context, a CNN architecture must be designed taking into account the device processing capacity [8]. In other words, besides providing a high accuracy rate, the CNN architecture must provide high detection throughput, with low memory demands. However, in the literature, in general, such factors are often provided individually [9].

The CNN can be implemented through different topology configurations, i.e. architectures, which are structured by layers. The number of used layers and their disposal impact the CNN complexity. In general, in the literature, to increase detection accuracy, authors often increase the CNN architecture size, by adding more layers [10]. As a result, popular proposed architectures may need up to 16 Gigabytes of memory, while processing only 4.26 frames/sec in a high-end GPU [11] [12].

In recent years, several authors have proposed CNN architectures suited for processing-constrained devices [13]. In general, authors aim at decreasing the proposed CNN architecture memory demand, while having minimal accuracy

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

impact [14]. However, a smaller CNN architecture does not necessarily increase the detection throughput. This because each CNN layer type introduces a related processing demand, consequently, besides aiming lower memory demands one must also evaluate the detection throughput [15]. However, surprisingly, for increasing the processing throughput, authors often only aim at decreasing the CNN memory size, without taking into account the layers specificity or the application-specific requirements [16].

Therefore, real-time video content detection in resource-constrained devices remains an open challenge. Resource-constrained devices such as smartphones do not have enough memory available for run the state-of-the-art CNN architectures [17]. In addition, when applying a CNN architecture fitted for mobile devices, the detection throughput is often not given proper care. Nonetheless, the accuracy impact caused by porting the CNN architecture to such devices may render the CNN application unfeasible. Therefore, for proper CNN usage in resource-constrained devices, it becomes imperative that proposed techniques address CNN modifications in terms of memory, detection throughput, and accuracy.

This paper presents a threefold approach towards real-time video content detection for resource-constrained devices. First, we address the challenge of decreasing the memory needs of typical CNN architectures while also considering the detection throughput. Our experiments show that lower CNN memory demands do not necessarily increase the processing throughput. Consequently, our proposed technique address, besides accuracy, both memory and detection throughput at CNN architecture building phase. Second, we apply a frame sampling technique that increases the detection throughput by up to 10 times, with little or no effect on accuracy. Finally, to address the accuracy impact of CNN porting to resource-constrained devices, we present a classification verification technique. The proposed approach is able to increase the detection accuracy with no side-effect on processing neither throughput nor memory.

The remainder of this paper is organized as follows: Section 2 describes the work background on video content detection and CNN. Section 3 presents the proposed approach towards real-time video content detection in resource-constrained devices. Section 4 evaluates the proposed technique, while Section 5 presents a summary of related works. Finally, we conclude the work in Section 6.

II. Background

In this section, we set the background for video content detection. More specifically, we first introduce the video content detection process, then, we further describe the process through a CNN-based classification task.

A. Video Content Detection

Video content detection is typically achieved by three sequential modules, namely *frame extraction*, *frame classification*, and *frame label assignment*, as shown in Fig. 1 (*Video Content Detection*). First, in *frame extraction*, the video is split into a set of frames. The number of frames is defined according to the video coding process. For instance, in general, a video is made of 23.97 frames/sec [18]. Then, each extracted frame is individually classified during the *frame classification* process. During this phase, a label is assigned for each given frame by means of a classification technique, e.g. applying a CNN. Finally, according to the assigned label, the *frame label assignment* performs a pre-established process, e.g. alert the user.

To perform real-time video content classification, the frames must be classified at video content speed. In other words, the detection process must have a detection throughput higher than the video frame rate (e.g. higher than 23.97 frames/sec). However, besides having to classify each frame individually, resource-constrained devices like smartphones perform other tasks in parallel, such as playing the video, or execute other applications in the background. Consequently, the video content detection task must be able to use as few as possible the available resources. The demanded execution by the video content classification task is mainly caused by the frame classification module. This because the module must evaluate each extracted video frame to properly label them. To this end, the module relies on a classification technique, such as applying a convolutional neural network.

B. Convolutional Neural Network

The architecture of a CNN is inspired by the human visual cortex. Such a deep neural network has been successfully applied to solve different problems in the field of Pattern Recognition. As shown in Fig. 1, CNN is made of layers with different purposes, as follows:

- *Input layer*: it receives an image with a specific resolution depending on the CNN architecture. The input image is represented as a matrix of pixels with dimensions (*width* x *height* x *depth*), where *depth* is the number of color channels in the image.
- *Convolutional layer (ConvLayer)*: it convolves a set of learnable kernels over the image to generate different feature maps. The first convolutional layers of a CNN extract low-level features from the input image which are minor details usually related to contour and color information. The subsequent ConvLayers extract on top of that low-level information, middle and high-level features representing major details related to the shape of the objects in the image. Such different feature levels extracted using the ConvLayers provide a complete understanding of the whole image being processed. One may find CNN architectures with different number of ConvLayers in the literature.
- *Pooling layer*: it applies a non-linear function on portions of the image covered by a kernel with a predefined size. The main motivation is dimensionality reduction, decreasing the number of features while reducing the chance of overfitting. To this end, the kernel is convolved over the image, while a non-linear function returns a value based on *Max* or *Avg Pooling*. The former returns

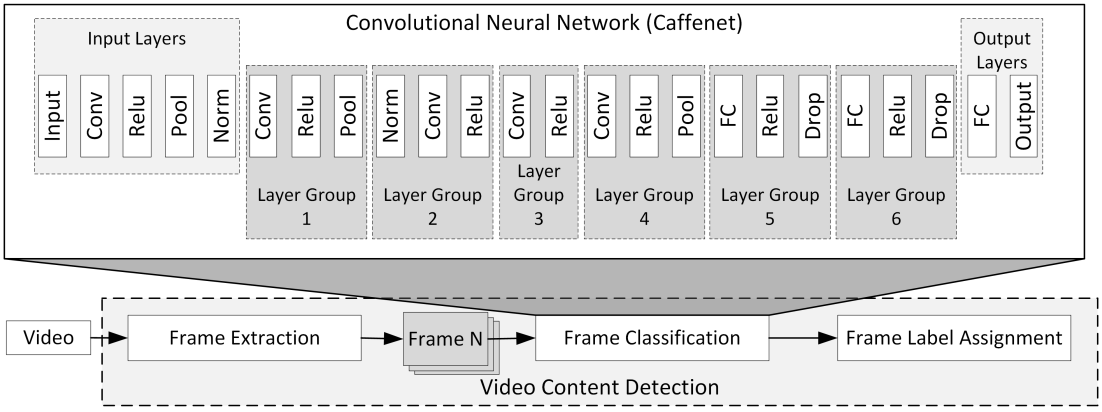

Fig. 1. Typical video content detection process.

the maximum value of the portion of the image covered by the kernel, while the later returns the average of all values.

- *Normalization layer*: it standardizes the inputs of a layer to accelerate the learning process.
- *ReLU layer*: it is an activation function that computes the output of a neuron. ReLU stands for rectified linear unit and it is the most used activation function for CNNs. The reason is that it can avoid that the gradient becomes zero, i.e. the well-known gradient vanishing problem. Formally, it is defined as $y = max(0,x)$.
- *Dropout layer*: it plays an important role in reducing the chance of overfitting by dropping out some units of the CNN during training. In other words, it avoids the co-dependency of neurons during training.
- *Fully Connected layer (FC)*: it is responsible to classify the image. To this end, the feature maps extracted by the convolutional part of the CNN are converted to a flattened structure (vector). The flattened structure goes through the FC part of the CNN to predict a score for each problem class.
- *Output layer*: it predicts a probability for each class.

It is worth to mention that the computational cost of the CNN model (memory and processing time) is usually directly related to the number of layers on it. Consequently, to apply CNN-based solutions on resource-constrained devices, one must find a compromise between computational cost and accuracy.

III. Real-Time Video Content Detection In Resource-Constrained Devices

To achieve real-time video content detection in resource-constrained devices our proposed technique is threefold: *frame sampling*, *real-time lightweight CNN*, and a *verifier* as shown in Fig. 2. The goal is to increase detection throughput, decrease memory demands, and maintain or even improve detection accuracy. The *frame sampling* technique aims at decreasing the number of frames that are going to be classified. To this end, the proposed approach samples the video frames as they are occurring, consequently decreasing processing demands. Then, the *real-time lightweight CNN* classifies the selected frames. The CNN, in turn, is built taking into account the accuracy, memory, and detection throughput rates.

Our proposal compounds the CNN architecture according to the layers processing and memory demands, while also taking into account the obtained accuracy. Thus, verifier module aims at improving the system accuracy with no additional processing demands. To this end, it applies classification output verification according to the CNN confidence values.

The next subsections describe in detail these three modules, including the proposed CNN building phase and the verifier module.

A. Frame Sampling

As described previously, a video is made by a sequence of frames (Section 2.1), for instance, 23.97 frames/sec. In general, video content detection approaches in the literature evaluate all frames, regardless of the application requirements. However, a video streaming application does not require the evaluation of all video frames, but only a subset of video frames can be evaluated, such as those which the frame content changes significantly, thus, reducing the processing demands.

In light of this, the *frame sampling* module samples the video frames which are going to be evaluated. Consequently, the processing demands can be significantly decreased, as not all frames will need to be evaluated by the classification approach. In order to select the video frames, the frame sampling can be implemented proactively or reactively. In the prior, the video frames are selected according to their content change. For instance, they can be evaluated only with respect to the KeyFrames from MPEG encoded videos [19]. In contrast, the reactive approach selects the video frames regardless of their content. For instance, select and evaluate after every N video frames. Hence, each frame selection technique must be selected according to the application demands and is context-specific.

Therefore, the *frame sampling* approach can significantly decrease the processing demands but introducing a detection delay and possible frame losses. In the prior, the user will only be alerted after a period of time, concerning to the prior frame

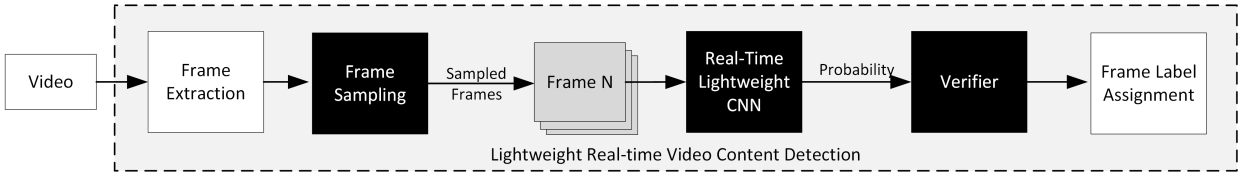

Fig. 2. Proposed real-time video content detection architecture for processing constrained devices.

occurrence. For instance, after N-1 frames have passed. In the former, the detection technique may lose some video frames, thus, not alerting the user. However, it is important to note that, in general, video content detection searches for content that occurs for several frames. For instance, a video content detection designed for finding X-rated content in videos, in such a case, that kind of frames occurs in several sequential frames, rather than in a single frame, hence, the proposed approach can properly leverage the frame sampling technique.

B. Real-Time Lightweight CNN

In general, CNN architectures are evaluated with respect to either accuracy, memory or throughput measures. Hence, in general, authors often perform CNN architecture modifications to improve a specific measure, e.g. accuracy, while impacting other measures, e.g. memory. However, for proper resource-constrained devices deployment, CNN architectures must provide high accuracies, with low memory requirements and a high detection throughput. Therefore, accuracy, memory and detection throughput must be evaluated together at the CNN building phase. As a result, the CNN architectures, i.e. the layers disposal, should be evaluated according to the target device resource availability and requirements. Consequently, the built CNN architecture will then be able to provide a real-time lightweight CNN proper for resource-constrained devices deployment.

In this work, we propose a novel CNN performance measure namely *lightweight*. The *lightweight* measure is computed as the sum of the normalized values of memory, detection throughput, and accuracy. The computation process takes into account the set of evaluated CNN architectures, and normalizes all obtained values, e.g. applying a Z-score normalization. Consequently, each performance measure has the same weight in the *lightweight* measure computation.

As a result, the *lightweight* measure comprises the set of desired CNN properties for proper deployment in resource-constrained devices. Therefore, the proposed lightweight measure enables one to properly evaluate CNN architectures for real-time CNN-based detection.

C. Verifier

In general, to achieve low memory demands and higher detection throughput, CNN architectures are modified and a tradeoff with accuracy is evidenced. However, accuracy degradation may render some applications unfeasible for deployment, even for resource-constrained devices. Consequently, despite decreasing the memory demands and increasing the detection throughput, the accuracy rates should remain similar, or even improve, to those obtained with the default CNN layer disposal. On the other hand, to increase detection accuracy, in general, one must perform more data processing, hence, decreasing detection throughput.

In the *verifier* module, we propose verify the CNN output confidence values. The *verifier* module accepts or not a CNN classification according to the CNN output confidence value. In other words, the *verifier* module checks whether the confidence value is higher than a given threshold value. Consequently, only high confidence classifications are accepted, thus, increasing the system accuracy in a lightweight manner. This because for each classification, only a simple threshold verification must be performed without requiring any additional computation (or CNN modification). Such a threshold value was experimentally defined considering different verifier operation points (error-reject tradeoff) as shown in Section IV.

Therefore, when applying the *verifier* module, the user can ensure that only highly confident frames were accepted and classified. As a result, false alarms can be reduced while processing demands remain low.

D. Discussion

The proposed real-time video content detection architecture for resource-constrained devices leverages the application settings in which a video content detection technique is typically employed. In general, such approaches are used in real-time applications, such as video streaming platforms. Consequently, we provide real-time detection in a threefold manner. First, the *frame sampling* selects only a subset of the video frames for evaluation. This approach significantly reduces the processing demands, as not all video frames are evaluated. Second, for providing a real-time lightweight CNN, we propose a novel performance measure technique namely *lightweight*. The measure takes into account the memory, detection throughput and accuracy for selecting the best CNN architecture for deployment. Finally, the *verifier* module accepts only highly confident classification, hence, increasing system accuracy with little or no performance impact. Therefore, the proposed approach, differently from related works, addresses real-time video content detection taking into account the resource-constrained nature of the devices in which it is going to be deployed.

IV. Experimental Evaluation

We present our proposal analysis by answering 3 research questions (RQ). For each research question, we present the evaluation criteria and the obtained results.

For evaluation purposes, we consider a scenario in which a resource-constrained device user aims to detect in real-time pornographic content in videos. We applied the well-known Pornography2k dataset [20], a dataset comprising 2 thousand videos, in which 1 thousand are pornographic and the remaining normal. For both video classes, the dataset comprises persons from different ethnicities and races in different activities. The default frame sampling periodicity is set at 0.3 frames/sec., similar to related works [21]. At total, the dataset is made of 1,376,037 video frames, extracted from 1,000 pornographic videos and 1,000 normal ones. For evaluation purposes the dataset was split into three parts: training, validation, and testing - each part comprising a proportion of 60%, 20% and 20%, respectively, for both pornography and normal videos. Consequently, each dataset comprises different videos, either normal or pornographic.

The CNN architectures are implemented and evaluated through Caffe [22] deep learning framework. Each architecture is trained through the training dataset and evaluated using the testing dataset. The final results are shown using the testing samples. Besides, each evaluated CNN architecture is trained for 50,000 epochs, with the learning rate set individually, by finding the best CNN parameters empirically [23]. Similar to related works, all input layers are composed by a 256x256 matrix size [24]. The CNNs were built in an Ubuntu 16.04 desktop equipped with an Intel i7, 16GB of memory, and an Nvidia Titan X GPU. The model which obtained the best results on validation dataset is used for evaluation on testing dataset.

A. RQ1: Does lightweight measure aid CNN building for resource-constrained devices deployment?

This RQ aims at evaluating whether the usage of the proposed *lightweight* measure (see Section 3.2) at the CNN architecture building phase can improve current state-of-the-art CNN architectures. To this end, the Caffenet [25] CNN architecture layout was tailored concerning the obtained *lightweight* measure. The goal is to create a custom-tailored CNN architecture which is built according to the *lightweight* measure. To achieve such a goal, the Caffenet architecture was split into 8 parts, each one made of specific layers as shown in Fig. 1. The CNN layers were grouped respecting their purposes; hence, each group performs a sequence of operations typically found in a CNN architecture. Consequently, the CNN size can be decreased by removing specific groups without damage the CNN architecture. The goal is to perform typical modifications made towards lightweight CNN architectures. In addition, the input and output layers are always used in the CNN architecture, while the other layer inside the groups 1 to 6 can be used or not. Due to data skewness, for the *lightweight* measure computation, each measure, accuracy, frames/sec (FPS), and throughput, was normalized using the Z-score normalization process [26].

Table I shows the obtained accuracy, memory requirements, detection throughput, and *lightweight* measure when the group layers are added or removed. It is possible to note a significant memory, accuracy, and throughput tradeoff, when layers are removed or added. The best *lightweight* measure CNN was obtained using groups 1, 2, 3 and 4. It was able to reach 89.30% of accuracy, with only 0.67% accuracy tradeoff to the most accurate CNN, while demanding only 1.21% of memory and achieving 270% more detection throughput. When comparing to the fastest CNN, it presented 123 fewer frames/sec detection throughput, while achieving 2.22% higher accuracy and requiring only 7.3 MB more memory.

TABLE I
LIGHTWEIGHT MEASURES OF CNN-BASED VIDEO CONTENT DETECTION TECHNIQUES.

Caffenet CNN Architecture Layout Groups						***Acc.***	***FPS***	***Mem. (MB)***	***Lightweight Measure***
1	2	3	4	5	6				
✓	✓	✓	✓	✓	✓	88.03	63.97	217	-0.41
	✓	✓	✓	✓	✓	**89.97**	81.37	746	-0.38
		✓	✓	✓	✓	88.92	91.54	743	-0.88
			✓	✓	✓	89.16	98.57	741	-0.68
				✓	✓	84.51	68.25	1200	-4.54
					✓	84.48	73.01	1100	-4.30
✓	✓	✓	✓	✓		89.47	197.21	153	1.59
✓	✓	✓	✓			**89.30**	**301.69**	**9**	**2.64**
✓	✓	✓				88.33	304.70	7.8	2.12
✓	✓					88.37	343.62	5.2	2.46
✓						87.08	**423.30**	**1.7**	2.38

Consequently, it is possible to note that the *lightweight* measure aids the CNN deployment for resource-constrained devices. The proposed measure enables to find the optimal CNN architecture, which improves the set of desired properties, accuracy, detection throughput, and memory usage, enabling the proper deployment of the CNN architecture for resource-constrained devices.

B. RQ2: Does frame sampling technique impact model accuracy?

This RQ aims at evaluating the model accuracy tradeoff caused by decreasing the number of frames used for training and evaluation purposes, which, as a result, improves detection throughput (see Section 3.1). To this end, the frame samples are chosen *reactively* for both training and evaluation. In other words, video frames are selected at specific time intervals, instead of choosing specific frames, such as KeyFrames. The reason is that the frame sampling aims at increasing the detection throughput. Therefore, choosing the frames *reactively* enables to further decrease the demanded processing. Consequently, the CNN architectures are trained and evaluated with respect to a chosen frame sampling periodicity.

Fig. 3 shows the relation between CaffeNet CNN accuracy, more specifically true-negative and true-positive rates, and frame sampling interval. It is possible to note that increasing the frame sampling interval introduces no significant effect on model accuracy. For instance, when using a 1.5-second frame sampling detection delay, as frames will be evaluated in higher time intervals. Thus, the frame sampling periodicity

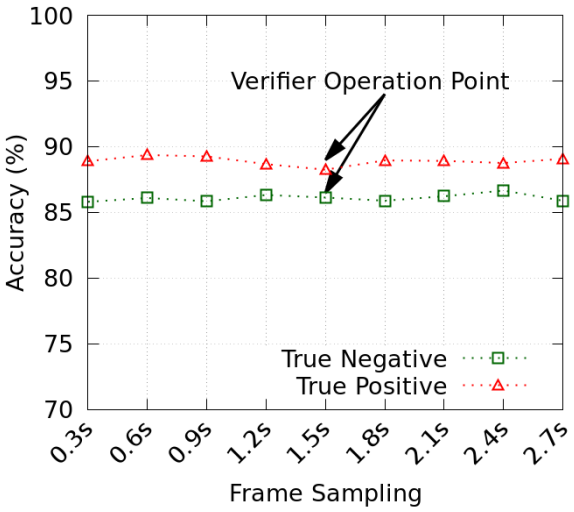

Fig. 3. CaffeNet CNN architecture accuracy and frame sampling tradeoff.

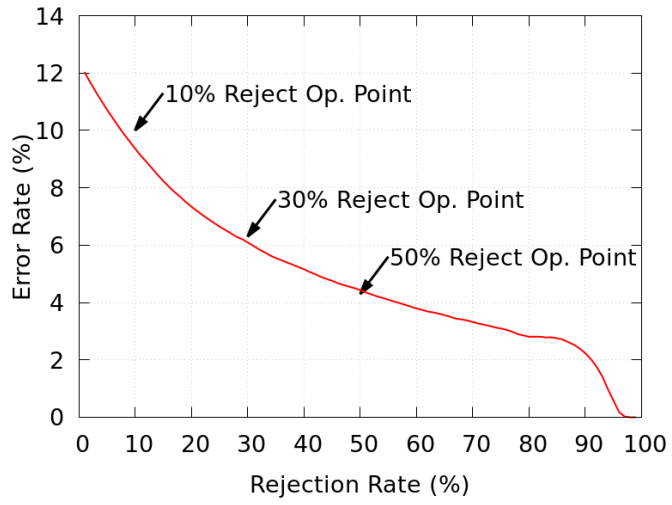

Fig. 4. CaffeNet CNN error-reject tradeoff applying verifier module with 1.5 second frame sampling interval.

must be selected according to the administrator needs, considering the applications-specific requirements, and the resource-constrained device capacity.

C. RQ3: Does verification technique aid at improving model accuracy?

This RQ aims at evaluating the verifier module improvement on accuracy (see Section 3.3). To this end, the CaffeNet CNN architecture with 1.5-second frame sampling interval was used (Fig. 3, *Verifier Operation Point*). Then, the CNN confidence values were used as a measure of confidence in frame classification. Consequently, the verifier module establishes whether the frame outcome should be accepted or not taking into account the confidence values. To find the verifier module operation points, the CRT [27] approach was used, which relies in class-specific thresholds for confidence computation.

TABLE II
CAFFENET ACCURACY IMPROVEMENT THROUGH THE PROPOSED VERIFIER MODULE.

Operation Point	*Accuracy*	*True-Positive (%)*	True-Negative
No Verifier	88.03	88.99	85.79
10% Rejection	91.01	91.97	88.81
30% Rejection	95.93	97.54	92.47
50% Rejection	97.66	99.42	94.02

Fig. 4 shows the CaffeNet CNN error-rate and rejection rate tradeoff. It is possible to note a direct relationship between the rejection and error rates. In other words, when increasing the rejection rate, one is able to decrease the system error rate, hence, further improving the system accuracy. Therefore, the verification module aids the CNN-based classification system to increase the obtained accuracy, with low or no processing demands, as the verification module only applies a simple threshold for the classification verification.

Table II further details the accuracy improvement of the verification module by applying the operation points marked in Fig. 4. It is possible to note a significant accuracy improvement with only 10% rejection rate, improving 2.98% and 3.02% the true-positive and true-negative rates respectively. In addition, by further increasing the rejection rate, for instance with a 30% rejection rate, the true-positive and true-negative rates further increases by 8.55% and 6.68%, respectively.

Therefore, the proposed verification technique enables to significantly improve detection accuracies with no significant processing tradeoffs. An application applying such a technique is able to verify whether the outcome of a lightweight CNN can be reliably accepted, hence, decreasing the ratio of false-positives, further increasing user reliability on the detection scheme.

D. Discussion

The design of a CNN-based detection scheme for resource-constrained devices must consider the tradeoffs that such porting introduces. The evaluation results have shown that the proposed lightweight measure aids the proper assessment of the introduced tradeoffs between memory, detection throughput, and accuracy. By leveraging such a measure, the administrator is able to properly establish which CNN architecture should be deployed in production. Nonetheless, the evaluation of the proposed frame sampling technique, in order to further increase the detection throughput, has shown that less frames can be used for evaluation and training purposes. Consequently, the administrator is able to significantly increase detection throughput, with little effect on accuracy. Finally, the evaluation of the proposed verification module has shown that one is able to employ the CNN confidence values as a measure of reliability to accept the lightweight CNN classification outcome. Hence, by using simple classification thresholds, the administrator can improve the detection accuracy.

Therefore, the evaluation results have shown that the proposed real-time video content detection architecture for pro-

cessing constrained devices is able to significantly improve detection throughput, accuracy, and memory demands.

V. Related Works

In general, proposed CNN architectures aim at either accuracy improvement, memory usage reduction or detection throughput increase. However, little or no effort has been conducted on Caffenet was made based on Alexnet, more specifically. It was built by a wrong Alexnet implementation. However, it is widely used in the literature as it presents reasonable better results than Alexnet architecture [28]. Finally, another widely used architecture was proposed by Szegedy et al., namely GoogleNet [29], also known as Inception, made by 22 layers. GoogleNet assumes that an optimal local sparse structure can be approximated and covered by available and ready densely components.

On the other hand, due to the memory limitations of mobile devices, several new CNN architectures were proposed towards smaller architectures [16]. For instance, Mobilenet [30], proposed by Google, applies separable in-depth convolutional functions to significantly decrease the number of used parameters, hence, decreasing the memory needs. Similarly, another CNN architecture, namely Squeezenet [31], also aims at decreasing the memory needs by decreasing the number of needed CNN parameters. Finally, ShuffleNet [32] decreases its memory size by applying two operations, the point to point convolutional and channel exchange, aiming at decreasing memory and processing needs while also maintaining the accuracy.

Despite presenting low memory needs, the detection throughput has also been chased by related works. For instance, YOLO [33] architecture, aimed at detecting image objects, is able to process in real-time, reaching up to 45 frames/sec. Similarly, in Fast R-CNN [34] architecture an object recognition approach is proposed by applying region-based convolutional networks. An enhancement is proposed by Faster R-CNN [35], which predicts the object boundaries according to its image position.

TABLE III
Lightweight measures of CNN-based video content detection techniques.

CNN Architecture	*Accuracy*	*Throughput (frames/sec)*	**Memory (MB)**
Alexnet	87.56	61.12	217
Caffenet	87.97	63.97	217
GoogleNet	91.51	18.29	48
Mobilenet	88.98	18.53	8.7
PVANet	83.88	10.79	292
Resnet	88.64	22.50	90
Shufflenet	86.76	37.82	3.5
Squeezenet	87.27	31.91	2.9
VGG	91.20	4.26	269
Proposed Approach	**89.30**	**301.69**	**9.0**

Table III shows the proposed approach performance with respect to the related works using the model which obtained the best *lightweight* measure (Table I). Our technique significantly improves memory demands, and detection throughput, even when compared to CNN architectures designed for such task, with little or no accuracy impact. Besides, we are able to improve further detection accuracy by applying the verifier module, and detection throughput by applying the frame sampling technique.

VI. Conclusions And Future Work

This paper has addressed the challenge of designing a CNN-based classification scheme for resource-constrained devices. To fulfill such a challenging task, our proposed architecture takes into account the desired properties from a lightweight CNN-based classification approach. In other words, we couple the memory requirements, detection throughput and accuracy during the CNN building and evaluation tasks. First, we introduce a *lightweight* measure which aids the administrator to properly establish the best CNN for resource-constrained devices. Then, by employing our frame sampling technique, we have shown that one is able to significantly improve detection throughput with little or no effect on accuracy. Finally, our verification module further improves the accuracy of the designed lightweight CNN, by applying a simple classification threshold according to the classification outcome. Consequently, our proposed approach significantly improved accuracy, memory demands, and detection throughput. For future work, we are going to port and evaluate the proposed approach performance in embedded devices. In addition, we plan to evaluate the proposed technique in other areas, such as image object recognition.

Acknowledgment

The authors thank CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for partial financial support (grants 430972/2018-0 and 306684/2018-2) and URCOP (Unidade de Repressão aos Crimes de Ódio e à Pornografia Infantil), a special division for pornography crackdown of Federal Police of Brazil for the support to the project.

References

[1] "How Many Phones Are In The World? 1 Billion More Mobile Connections Than People Worldwide: 2019." https://www.bankmycell.com/blog/how-manyphones- are-in-the-world. Accessed: jan. 2020.

[2] "Number of smartphone users worldwide from 2016 to 2021" https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. Accessed: jan. 2020.

[3] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Y. Choi and T. R. Faughnan, "Real-time human detection as an edge service enabled by a lightweight CNN". Proceedings - 2018 IEEE International Conference on Edge Computing, EDGE 2018 - Part of the 2018 IEEE World Congress on Services, Sep. 2018, 125–129.

[4] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network". Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Decem, 2016, 1874–1883. DOI:https://doi.org/10.1109/CVPR.2016.207.

[5] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, "Large-scale video classification with convolutional neural networks". Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Sep. 2014, 1725–1732.

[6] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks". Communications of the ACM. 60, May 2017, 84–90. DOI:https://doi.org/10.1145/3065386.

[7] A. Plieninger, "Deep Learning Neural Networks on Mobile Platforms", 2015.

[8] H. Ye, Z. Wu, R. Zhao, X. Wang, Y. Jiang and X. Xue, "Evaluating Two-Stream CNN for Video Classification Categories and Subject Descriptors", 2015, 435–442.

[9] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, L. Wang, G. Wang, J. Cai and T. Chen, "Recent Advances in Convolutional Neural Networks", Dec. 2015.

[10] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", Sep. 2014, 1–14.

[11] M. Imani, D. Peroni, Y. Kim, A. Rahimi and T. Rosing, "Efficient neural network acceleration on GPGPU using content addressable memory". Proceedings of the 2017 Design, Automation and Test in Europe, 2017, 1026–1031. DOI:https://doi.org/10.23919/DATE.2017.7927141.

[12] H. Park, D. Kim, J. Ahn and S. Yoo, "Zero and data reuse-aware fast convolution for deep neural networks on GPU". 2016 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS, 2016, 1–10. DOI:https://doi.org/10.1145/2968456.2968476.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and C. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks". Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Dec. 2018, 4510–4520.

[14] A. Lavin and S. Gray, "Fast Algorithms for Convolutional Neural Networks". Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, 4013–4021.

[15] J. Wu, C. Leng, Y. Wang, Q. Hu and J. Cheng, "Quantized convolutional neural networks for mobile devices. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition", Decem, 2016, 4820–4828. DOI:https://doi.org/10.1109/CVPR.2016.521.

[16] D. Anisimov and T. Khanova, "Towards lightweight convolutional neural networks for object detection". 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS, Oct. 2017.

[17] N. Liu, L. Wan, Y. Zhang, T. Zhou, H. Huo and T. Fang, "Exploiting Convolutional Neural Networks With Deeply Local Description for Remote Sensing Image Classification". IEEE Access. 6, Jan. 2018, 11215–11227. DOI:https://doi.org/10.1109/ACCESS.2018.2798799.

[18] Y. Kuroki, T. Nishi, S. Kobayashi, H. Oyaizu and S. Yoshimura. "A psychophysical study of improvements in motion-image quality by using high frame rates". Journal of the Society for Information Display. 15, 1, 2007, 61. DOI:https://doi.org/10.1889/1.2451560.

[19] M. Perez, S. Avila, D. Moreira, D. Moraes, V. Testoni, E. Valle, S. Goldenstein and A. Rocha, "Video pornography detection through deep learning techniques and motion information. Neurocomputing". 230, Mar. 2017, 279–293. DOI:https://doi.org/10.1016/j.neucom.2016.12.017.

[20] D. Moreira, S. Avila, M. Perez, D. Moraes, V. Testoni, E. Valle, S. Goldenstein and A. Rocha, "Pornography classification: The hidden clues in video space–time. Forensic Science International". 268, Nov. 2016, 46–61. DOI:https://doi.org/10.1016/j.forsciint.2016.09.010.

[21] J. Y. H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga and G. Toderici, "Beyond short snippets: Deep networks for video classification", 2015.

[22] "Caffe - Deep Learning Framework: 2019". https://caffe.berkeleyvision.org/. Accessed: 2019-09-12.

[23] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao and S. Yan, "CNN: Single-label to Multi-label". 6, 1, 2014, 1–14. DOI:https://doi.org/10.1109/TPAMI.2015.2491929.

[24] K. Nogueira, O. A. B. Penatti and J. A. Santos, "Towards better exploiting convolutional neural networks for remote sensing scene classification". Pattern Recognition, 2017. DOI:https://doi.org/10.1016/j.patcog.2016.07.001.

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding". arXiv preprint arXiv:1408.5093, 2014.

[26] A. Jain, K. Nandakumar and A. Ross, "Score normalization in multimodal biometric systems". Pattern Recognition, 2005. DOI:https://doi.org/10.1016/j.patcog.2005.01.012.

[27] G. Fumera, F. Roli and G. Giacinto, "Rapid and Brief Communication Reject option with multiple thresholds". 33, 2000, 2099–2101.

[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell and U. C. B. Eecs, Jia - Caffe - Convolutional Method for Fast Feature Embedding, 2014.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions". Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Oct. 2015, 1–9.

[30] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", Apr. 2017.

[31] F. N. Iandola, S. Han, M. W, Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, Feb. 2016.

[32] X. Zhang, X. Zhou, M. Lin and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices". Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Dec. 2018, 6848–6856.

[33] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection". Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Decem, 2016, 779–788. DOI:https://doi.org/10.1109/CVPR.2016.91.

[34] R. Girshick, "Fast R-CNN". Proceedings of the IEEE International Conference on Computer Vision. Inter, 2015, 1440–1448. DOI:https://doi.org/10.1109/ICCV.2015.169.

[35] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real - Time Object Detection with Region Proposal Networks". IEEE Transactions on Pattern Analysis and Machine Intelligence. 39, 6 ,2017, 1137–1149. DOI:https://doi.org/10.1109/TPAMI.2016.2577031.