Classification of vehicle make and model with MobileNets

Renan B. de C. Wille, Ricardo Carnieri, Gustavo B. Borba and Daniel R. Pipa.

***Abstract*— Video monitoring generates large amounts of raw data from which relevant information can be extracted using image processing techniques. When these cameras are used in tolls or for traffic monitoring it is interesting to acquire characteristics like color, license plate, make and model of the vehicles passing by. This work proposes the use of a recent class of deep learning models called MobileNets on the task of retrieving the make and model information of vehicle images. The usage of these types of models can lower computational cost and improve classification accuracy. The CompCars dataset is used to assess the accuracy of the proposed method on the task of retrieving cars make and model. Results show an improvement of 2.5% on the top-1 accuracy if compared to that reported in the extended CompCars work. Moreover, it is shown that, by means of the variations of MobileNets architectures, one can obtain the desired trade-off between complexity (computational cost) and accuracy. This is an effective approach to set up the system to match the application's requirements.**

***Keywords*— Deep learning, CompCars, MobileNets, vehicle make and model**

I. Introduction

Surveillance cameras are being used in many applications for monitoring human behavior. In streets and public roads, cameras are on tolls or for traffic monitoring, for example. These are scenarios on which they extract images that contain the raw data about the vehicles that are circulating. Through the usage of image processing techniques, characteristics of the vehicles like make and model, color, license plate can be extracted. This information can be used for many purposes in intelligent transportation systems. For example, from the visual identity of the vehicle, obtained by means of the combination of the extracted features, one can verify the vehicle against an expected identity and check for vehicle cloning or appearance changes. Also, by automatically detecting vehicle make and model, it is possible to index them in parking systems.

In order to select the relevant information, deep neural networks (DNNs) are standing out in relation to other image processing methods. By using a recent class of DNNs called MobileNets [1] this research aims to improve the recognition accuracy and lower the computational load of the make and model classification task proposed by [2] on the CompCars dataset created in that same work.

The rest of the paper is organized as follows. First, the related literature is reviewed. Next, the materials and methods of how the research was conducted are explained in the approach section. After that, the experiments and results are listed and explained, and finally, the article is concluded.

II. Related Work

Several methods have been proposed to classify the vehicle make and model from images. We can organize them into part-based methods and global methods. Part-based methods try to infer the make and model from distinctive vehicle parts, such as the logo, model name, and headlights, which must first be segmented from the full image. Global methods, on the other hand, are segmentation-free and calculate features of the entire vehicle encoding vehicle silhouette and texture.

Llorca et al. [3] proposed a method to recognize the vehicle logo. They use an Automatic License Plate Recognition (ALPR) system to first find the license plate location, and then they assume that the logo is located above it. This area is searched by means of a sliding window approach generating many regions that are described by Histogram of Gradients (HOG) and classified by a linear Support Vector Machine (SVM). The authors also extend the research to try to find vehicle models [4]. In that paper, they limit the model search to only those of the previously recognized brand. Geometric constraints and HOG are used to describe the regions containing the model name, which are then classified.

Another work that tries to recognize vehicle logos is Huang et al. [5] that also use an ALPR system to first locate the license plate. Then they assume that the vehicle logo is located above it. After that a coarse segmentation of the region is extracted and classified with a convolutional neural network, assuming that the classifier is translation invariant.

Note that by using parts of the vehicles for identification as the described methods, if the segmentation step of the region of interest does not work, the prediction will likely be erroneous. The following methods use entire information of the vehicle avoiding this problem.

Sochor et al. [6], automatically extract 3D bounding boxes of vehicles from surveillance images. These 3D boxes allow them to identify side, roof, and front/rear side of vehicles. This information is used to create a normalized image of the vehicle which, together with the encoded 3D bounding box and the encoded viewpoints are used as inputs to a convolutional neural network, that classifies the vehicle.

Nazemi et al. [7] proposed a system to classify 10 classes of make and model. From a frontal image of the vehicle, which can be in different viewpoints, they extract the position of the vehicle by classifying image patches with an SVM and HOG searching by the existence or not of a vehicle in each patch. Next, they join the positive patches into a region containing the vehicle, from which they extract features that should be invariant to illumination changes, viewpoint, and

noise. They use the densely sampled scale invariant feature transform (SIFT) [8] features that are encoded with sparse feature coding techniques based on Bag of Words(BOW) to describe the image and then an SVM is used to classify them.

Yang et al. [2] proposed a dataset with sufficient proportions for training DNNs called CompCars, in which there is a selected collection of images used for fine-grained classification of vehicle make and model. For this task, in an extension of the article available on arxiv [9] they open-sourced a model called GoogLeNet_cars and it can predict 431 classes of vehicle make and models from entire images containing the vehicles.

III. Approach

In this paper, we show that the classification accuracy on the CompCars fine-grained task in recognizing vehicle make and model can be improved while lowering computational costs by using the so-called MobileNets. When compared with the result of the fine-tuned GoogLeNet(GoogLeNet_cars) model described in [9].

The CompCars dataset contains two types of vehicles images: web-nature that are images collected from forums, public websites, and search engines; and surveillance-nature images collected from surveillance cameras. In particular, the web-nature set contains 163 car makes with 1716 car models, with a total of 136726 images. These pictures contain the entire vehicles in various viewpoints and inserted into different backgrounds. For the fine-grained classification task, these makes and models were combined into a subset of 431 make-model classes, with vehicles of the same model type but produced in different years assigned to same class. This subset contains 52083 images and is divided into 70% (36456 images) for training, referenced hereinafter as CompTrain, and 30% (15627) for testing, referenced hereinafter as CompTest. Examples of images from this dataset can be viewed in figure 1.

Fig. 1. Sample images from CompCars dataset [2]. Images were obtained from the web and depict vehicles under various viewpoints.

GoogLeNet_cars is a GoogLeNet [10] model that was fine-tuned on the CompTrain set. Meaning that first the network GoogLeNet was trained on ImageNet [11], then by using the pre-trained weights as an initialization and a lower learning rate this model is trained on the CompTrain set. This procedure, called fine-tuning, is normally used to save power and computational resources. The original GoogLeNet network contains approximately $1.59\text{x}10^9$ multiply-add operations and replaces the standard fully-connected layers at the end with a simple global average pooling that reduces the total number of parameters compared to other networks. We have chosen to compare with this model, first because the database is available, second because the model is open sourced, which allowed us to test it locally and validate the result of [9] and third because this was the only model found, that followed the same approach and had comparable results when this research was conducted.

In this work, we propose the use of MobileNets [1] on the CompCars task. They are a recent class of convolutional neural networks that, by means of depthwise separable convolutions, achieve lower computational cost during inference. Depthwise separable convolutions are a form of factorized convolutions; they separate the filtering and the combining operations of the standard convolutions. First, the depthwise convolution applies a single filter to each input channel. Then the pointwise filter, by applying a 1x1 convolution, combines the outputs. These models already have a reduced number of parameters, but hoping to have even faster models, Howard et al. [1] propose two values, width multiplier and resolution multiplier, that can be further tuned so that the user can choose networks with even smaller sizes, though at the cost of reduced accuracy. They have open-sourced 16 models pre-trained on ImageNet, obtained by sweeping these two parameters. The network trained on ImageNet with the highest accuracy uses resolution of 224 by 224 pixels and width multiplier of 1.0 and has approximately $569\text{x}10^6$ multiply-add operations.

IV. Fine-tuning

To fine-tune the proposed models we implemented the training process in Python using the Keras [12] framework with the Tensorflow [13] back-end. What motivated these choices was the ease-of-use of the Keras application programming interface and because the pre-trained MobileNet models were available for Tensorflow. For each pre-trained model we used, the last layer was replaced to reflect the number of expected classes. The MobileNet architecture we used, with width multiplier of 1.0 and input size of 224 by 224, can be seen in Table I.

To continue training a Keras model an image generator was built to open, pre-process, and provide the images of the CompTrain dataset to the model. To augment the dataset and mitigate the model overfitting on the data the images used during training are first resized to 256 by 256 pixels and then randomly cropped to 224 by 224 pixels. A random horizontal flip was also used for additional data augmentation. Finally, the generator is responsible for delivering the number of samples expected to fit the memory of the used hardware. The batch size contains 32 images that are sampled cyclically from the CompTrain set.

The training was made during 71 epochs (one epoch means sampling at every sample at least once from the dataset). This number was chosen because the GoogLeNet_cars model, according to the available solver, was trained with the Caffe [14] framework during 10000 iterations with 256 images for each iteration, that is approximately 70.22 epochs on the CompTrain set, which was rounded to 71 as the generator

TABLE I
SIMPLIFIED MOBILENET 1.0 224 ARCHITECTURE USED TO FINE-TUNE ON COMPCARS, WHERE CONV MEANS CONVOLUTION, S1 AND S2 ARE THE STRIDES SIZES AND DW MEANS DEPTHWISE TYPE. NOTE THAT THE LAYERS IN BOLD WERE REPLACED FOR THE FINE-TUNING PROCESS.

Type/Stride	Filter Shape	Input Size
conv/s2	3 x 3 x 3 x 32	224 x 224 x 3
Conv dw/s1	3 x 3 x 32 dw	112 x 112 x 32
Conv/s1	1 x 1 x 32 x 64	112 x 112 x 32
Conv dw/s2	3 x 3 x 64 dw	112 x 112 x 64
Conv/s1	1 x 1 x 64 x 128	56 x 56 x 64
Conv dw/s1	3 x 3 x 128 dw	56 x 56 x 128
Conv/s1	1 x 1 x 128 x 128	56 x 56 x 128
conv dw/s2	3 x 3 x 128 dw	56 x 56 x 128
Conv/s1	1 x 1 x 128 x 256	56 x 56 x 128
Conv dw / s1	3 x 3 x 256 dw	28 x 28 x 256
Conv / s1	1 x 1 x 256 x 256	28 x 28 x 256
Conv dw / s2	3 x 3 x 256 dw	28 x 28 x 256
Conv / s1	1 x 1 x 256 x 512	14 x 14 x 256
5x Conv dw / s1	3 x 3 x 512 dw	14 x 14 x 512
Conv / s1	1 x 1 x 512 x 512	14 x 14 x 512
Conv dw / s2	3 x 3 x 512 dw	14 x 14 x 512
Conv / s1	1 x 1 x 512 x 1024	7 x 7 x 512
Conv dw / s2	3 x 3 x 1024 dw	7 x 7 x 1024
Conv / s1	1 x 1 x 1024 x 1024	7 x 7 x 1024
GlobalAveragePooling2D	**Pool 7 x 7**	**7 x 7 x 1024**
FC / s1	**1024 x 431**	**1 x 1 x 1024**
Softmax	**Classifier**	**1 x 1 x 431**

needs an integer number of epochs. The learning rate schedule starts at 0.002 and after the middle of the training it is dropped to 0.0002. This value was chosen based on the batch size and so that the model started to learn right away. We used Stochastic Gradient Descent with a momentum of 0.9, which is the same optimizer used for training GoogLeNet_cars in [9]. A diagram detailing the process of fine-tuning can be seen in figure 2.

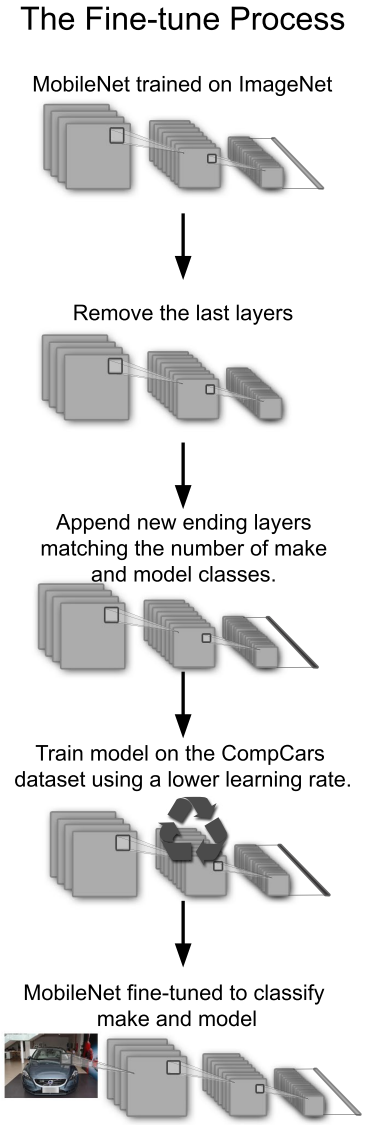

Fig. 2. Diagram detailing the fine-tuning process. Note that we do not train the model from the beginning, we start from the pre-trained weights.

V. EXPERIMENTS AND RESULTS

We performed seven tests to examine the results of some of the available models and inspect the influence of the different parameters. The metric used to compare the results was the accuracy (the number of right predictions divided by the total amount of samples). Where we consider in top 1 if the prediction with the highest probability was correct and on top 5 if the correct prediction is listed on the 5 highest probabilities. First, the input resolution of the MobileNets models was set to 224 by 244 and the width multiplier was changed from 1.0 to 0.25 in steps of 0.25. The results are listed in Table II. Note that the accuracy drops as expected when a smaller model is used. Also, the MobileNet-1.00-224 model has 2.5% higher accuracy than GoogLeNet cars while using fewer parameters. But compared with the results of the article of MobileNet [1] the drop in accuracy was lower than the one reported on ImageNet, possibly because the CompCars dataset has fewer classes and fewer images than ImageNet.

We then experimented with different values for the resolution multiplier while keeping the width multiplier set to 1.0. The input resolutions used were 224 by 224, 192 by 192, 160 by 160 and 128 by 128. To train these networks a secondary change was needed: since the original resized resolution was 256 by 256 pixels for the 224 by 224 cropped input, the input was proportionally resized for the smaller resolutions. The results are listed in Table III. The recall drops faster than it did for width parameter presumably because, as the image gets smaller, information like the logo of the vehicle loses resolution and becomes more blurred, making the classification rely more on the silhouette of the vehicle. Figure 3 resumes the results of all the fine-tuned models.

During the analysis of the highest accuracy model (MobileNet-1.00-224) we checked the mistakes made to verify if the model errors were making sense. We list in the Table IV the 3 classes that were most commonly mistaken. We notice that normally the mistakes occur between the same make but end up having the wrong models. The first two mistakes occur by the difference between the sedan and hatchback versions, it seems that the model struggles into differentiating the back of the car maybe because during training, some vehicle viewpoint

TABLE II

TESTS SWEEPING THE WIDTH MULTIPLIER PARAMETER OF THE FINE-TUNED MOBILENET AND COMPARING THE RESULT WITH GOOGLENET_CARS. NOTE THAT THE RECALL DROPS AS THE NETWORK GETS SMALLER AND THAT MOBILENETS CAN ACHIEVE BETTER RESULTS ON THE SAME DATASET.

Model	Accuracy TOP 1	Accuracy TOP 5
GoogLeNet_cars	91.2%	98.1%
MobileNet 1.00 224	**93.7%**	**98.8%**
MobileNet 0.75 224	92.2%	98.4%
MobileNet 0.50 224	89.8%	97.8%
MobileNet 0.25 224	79.5%	94.7%

TABLE III

TESTS SWEEPING THE RESOLUTION MULTIPLIER PARAMETER OF THE MOBILENET TO COMPARE WITH GOOGLENET_CARS. NOTE THAT THE RECALL DROPS ARE GREATER THAN IT WAS FOR THE WIDTH PARAMETER.

Model	Accuracy TOP 1	Accuracy TOP 5
GoogLeNet_cars	91.2%	98.1%
MobileNet 1.00 224	**93.7%**	**98.8%**
MobileNet 1.00 192	92.5%	98.5%
MobileNet 1.00 160	83.9%	95.6%
MobileNet 1.00 128	54.2%	76.7%

isn't available and the model can't generalize for this expected position. For the third mistake, analyzing the images of the dataset, we can view that for these two different classes there are vehicles that are visually similar and probably the DNN model isn't differentiating the nuances. Figure 4 contains samples of the misclassified images and its similar vehicle models on the misclassified class.

To also know the approximated inference time in a known embedded platform, we built Tensorflow for a Raspberry Pi 3 Model B that features 1 GB of RAM and a Quad Core 1.2Ghz ARM Cortex A53 CPU. By using the same Keras framework

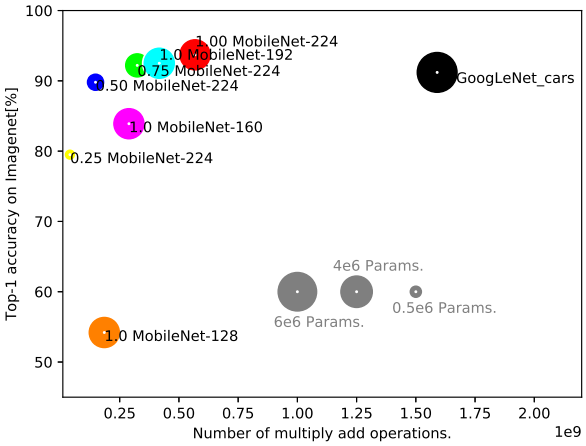

Fig. 3. Results of the fine-tuned models for the CompCars dataset. The plot presents Top-1 accuracy versus the number of multiply-add operations used during inference. The size ot the circles represent the number of parameters of each network. Some configurations of the MobileNets clearly overcomes GoogleLeNet_cars on this task, presenting even higher accuracy for a considerable lower number of multiply-add operations.

TABLE IV

TO CHECK WHAT COULD BE MADE TO IMPROVE PREDICTION WE LISTED THE MOST COMMON MISTAKES MADE BY MOBILENET-1.00-224 MODEL.

Ground Truth	Predicted	Num. Mistakes
Ford-New Focus Sedan	Ford-New Focus Hatchback	11
BAW-E Series Sedan	BAW-E Series Hatchback	8
Volvo-V40	Volvo-S40	8

Fig. 4. Left: images that the MobileNet-1.00-224 model misclassified. Right: similar vehicle models encountered on the misclassified class.

we made a sample of 500 images and used the fine-tuned models to infer their classes. The time of each inference was kept and at the end of the process, a mean time was calculated. The results are shown in Table V. We note that by using only the CPU the best accuracy model takes almost one second to predict one image. Also, changing the width multiplier reduces more the number of parameters, as can be viewed on figure 3, and by consequence, it has a greater impact over reducing inference time than by varying the resolution multiplier.

TABLE V

TO KNOW THE APPROXIMATED INFERENCE TIME ON AN EMBEDDED PLATFORM WE USED A RASPBERRY PI 3 MODEL B HARDWARE. THIS TABLE SHOWS THE MEAN TIME OF THE PREDICTION OF 500 RANDOMLY SAMPLED IMAGES FOR EACH FINETUNED MODEL.

Model	Mean Inference Time (ms)
MobileNet 1.00 224	846
MobileNet 0.75 224	796
MobileNet 0.50 224	490
MobileNet 0.25 224	282
MobileNet 1.00 192	793
MobileNet 1.00 160	613
MobileNet 1.00 128	443

VI. CONCLUSIONS

The goal of this research was to evaluate the MobileNets convolution neural networks on the CompCars vehicle make and model classification task. To validate if it could, even with a lower number of parameters, achieve better results on the

same task when compared with GoogLeNet_cars. By using the fine-tuning process described, it is possible to achieve 2.5% higher accuracy on the same task by using the best MobileNet configuration. Also, by sacrificing some accuracy, faster configurations like MobileNet-1.00-192 or MobileNet-0.75-224 are available, which are still slightly better than GoogLeNet_cars on the same task. Further work can be done on training the network; since no hyperparameter optimization was performed, better results might be achieved with different configurations like changing the optimizer or varying the learning rate.

REFERENCES

[1] 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," *ArXiv*, p. 9, 2017.

[2] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, no. 978. IEEE, jun 2015, pp. 3973–3981.

[3] D. F. Llorca, R. Arroyo, and M. A. Sotelo, "Vehicle logo recognition in traffic images using HOG features and SVM," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, no. Itsc. IEEE, oct 2013, pp. 2229–2234.

[4] D. F. Llorca, D. Colas, I. G. Daza, I. Parra, and M. a. Sotelo, "Vehicle model recognition using geometry and appearance of car emblems from rear view images," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, oct 2014, pp. 3094–3099.

[5] Y. Huang, R. Wu, Y. Sun, W. Wang, and X. Ding, "Vehicle Logo Recognition System Based on Convolutional Neural Networks With a Pretraining Strategy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1951–1960, aug 2015.

[6] J. Sochor, A. Herout, and J. Havel, "BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3006–3015, 2016.

[7] A. Nazemi, M. J. Shafiee, and Z. Azimifar, "On road vehicle make and model recognition via sparse feature coding," in *2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP)*. IEEE, sep 2013, pp. 436–440.

[8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[9] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," *arXiv*, 2015.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[12] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: ACM, 2014, pp. 675–678.