Fine-grained Cars Recognition using Deep Convolutional Neural Networks

Franklin Oliveira
Computing Center
Federal University of Pernambuco
Recife, Brazil
flso@cin.ufpe.br

Arianne Macena
Computing Center
Federal University of Pernambuco
Recife, Brazil
asm7@cin.ufpe.br

Otávio Kamel
Computing Center
Federal University of Pernambuco
Recife, Brazil
oackb@cin.ufpe.br

Wesley Souza
Computing Center
Federal University of Pernambuco
Recife, Brazil
ws@cin.ufpe.br

Nicksson Freitas
SiDi
Recife, Brazil
nickssonarrais@gmail.com

Tiago Vinuto
SiDi
Recife, Brazil
t.vinuto@sidi.org.br

***Abstract*—Population growth and the high concentration of vehicles on urban roads have negatively impacted urban mobility and the global environment since the primary transportation modes occupy a lot of space on the streets and are one of the main polluting gas emitters. In this context of inefficient urban mobility and unsustainability, the Intelligent Transportation Systems (ITS) aims to solve or minimize urban traffic issues. ITS are also widely used in applications focused on traffic safety, such as vehicle recognition related to a traffic or law violation. For this task, the fine-grained vehicle classification technique is mainly used by computer vision and deep learning advances. However, identifying vehicles by the model can be problematic because the same vehicle can be easily misclassified when observed from different perspectives, with different colours, or by similar models. Knowing these inherent issues from vehicle recognition tasks, Deep Convolutional Neural Networks (DCNNs) are commonly used due to their ability to extract features from images. In that regard, the goal of this paper is to evaluate some state of art DCNNs architectures, conducting experiments with three different datasets to identify which architectures have the best performance metrics in the refined car classification task within ITS context.**

I. Introduction

Due to the population exponential growth, the number of vehicles on public roads has increased significantly, causing issues such as intense traffic flow with heavy traffic jams, high traffic infractions and vehicle theft rates, particularly in developing countries such as Brazil, which had more than 18.5 millions traffic infractions and approximately 160 thousand vehicle theft cases in 2020 [1]. Faced with these issues, vehicle monitoring systems are required for a variety of purposes, including traffic flow control, vehicle identification with traffic infractions and even tracking stolen vehicles [2].

In this scenario, Intelligent Transportation Systems (ITS) play an important role in transportation management, security, and other issues, since there are applications that use various computer vision techniques for a wide range of ITS-related purposes, such as object detection and tracking, and image classification. In this context, the fine-grained vehicle classification task appears to recognize a vehicle by its make and model, and can be used in ITS that need to identify vehicles for the main purpose of recovering associated traffic violations and tracking stolen vehicles [3].

Furthermore, fine-grained vehicle classification is more complex than traditional classification tasks because it requires a deeper understanding of the vehicle domain to differentiate objects that may be very similar in a macro view. In this way, the main challenge of this task is to be able to distinguish vehicles of different make and model that appear to be very similar, as well as to be able to identify the same vehicle with variations in views and colors [4].

Faced with these challenges, the issues inherent in the fine-grained vehicle classification task have been extensively studied and experimented in works like [5] [6] [7] [8], where the authors explore approaches with DCNN architectures, demonstrating state-of-the-art results even with the complexity of the problem.

Given these fine-grained vehicle classification problems and the widespread usage of DCNNs to solve this task, this paper aims to analyze DCNN architectures, considering aspects such as model generalization and computational performance, being able to be deployed on edge devices for real-time road monitoring in the ITS context, for example. Therefore, this paper addresses a comparison between some DCNNs architectures, trained on three datasets, based on the generalization evaluation from machine learning metrics (accuracy, precision and recall) and computational performance through training and image inference time.

The main contributions of this paper is the evaluation and analysis of neural networks for car recognition task in the ITS context using state-of-the-art DCNNs architectures, comparing the results among the most used datasets for vehicle classification in the literature [3] and introducing a more recent dataset [8].

The next sections are organized as follows: in Section II, the relevant related works are presented. Section III describes the methodology in detail - more specifically: the datasets, experimented DCNN architectures, data augmentation techniques and evaluation metrics. The details of the experiments are given in Section IV. The results and their respective discussions are provided in Section V. Finally, conclusions are discussed in Section VI.

II. Related Works

Due to inherent fine-grained vehicle classification difficulties, this task has been widely addressed in the community [5] [6] [7]. The main issue in this context is how two or more vehicles may have high similarity, making it difficult to distinguish even for the human brain. Current researches are most concerned with developing technologies that can differentiate vehicles by make and model [9], requiring a vast knowledge of existing cars and the ability to identify them from various perspectives. Particularly, several recent studies focused on the fine-grained vehicle classification task use DCNN as the main approach, owing to their ability to extract features from images. Some of these more relevant studies will be described below.

Accordingly, Hassan *et al.* [10] evaluated the performance of recent DCNNs for vehicle make model recognition (VMMR) using accuracy as the main metric. The research implemented transfer learning to reduce the training time, data augmentation techniques to improve the models' generalization, and k-fold cross-validation to better validate the data results. All these methods were applied to improve the models' accuracy, using the Cars-196 dataset for training and the VMMR-db51 dataset for testing. The highest accuracy achieved was 93.96% in Cars-196 and 70% in VMMR-db51, using DenseNet201.

Sánchez *et al.* [3] evaluated state-of-the-art deep learning models (ResNet-50 and InceptionV3) performing the fine-grained vehicle classification task by exploring various training techniques. They used 3 different datasets to create correspondences between classes of CompCars, VMMR-db, and Frontal-103 to mitigate biases. Only InceptionV3 results were shown and they reached 78.47% for model classification.

Additionally, Ma *et al.* [5] proposed changes in the max pooling layers, improving make-model car classification task, and they compared this change to multiple CNN architectures (DenseNet161, VGG16, and ResNet-152) using the Cars-196 dataset, reaching an accuracy of 97.89% using ResNet-161. Complementarily, Kuhn and Moreira [8] also include a comparison of some CNNs (InceptionV3, ResNet-50, KNN InceptionV3-S, KNN ResNet50-S), using a different dataset made up of popular vehicles from Brazil (BRcars). They reached 82% precision and 79% recall with Inception-V3.

In contrast, Najeeb *et al.* fine-tuned seven architectures (AlexNet, SqueezeNet, ShuffleNet, GoogleNet, ResNet-18, MobileNet-v2, and Inception-v3) to improve the accuracy in classifying vehicles by make and model using their own dataset (THS-10 Dataset), generated from the collection of images with a camera installed on a bridge, capturing the traffic of a road [7]. The dataset has 4,250 images and 10 classes and InceptionV3 achieved the highest accuracy (97.4%) in this settup.

III. Methodology

Aiming to extend and converge the fine-grained vehicle classification studies and based on the works presented previously, this research discusses which state-of-the-art DCNNs should be more reliable and efficient in this task. The possibility of implementing a comparative study using multiple architectures was investigated [3] [10].

Firstly, the DCNNs architectures were selected. During the training, a pipeline that applies an average grouping in the spatial dimensions was implemented, transforming it into a single vector [8]. It then proceeds to two fully connected layers with 1024 and 512 channels respectively, followed by a dropout layer to avoid overfitting with a drop rate of 0.5. The final step is to apply a softmax function in the network output. In short, the common structure is: CNN layers $\rightarrow$ Global Average Pooling layer $\rightarrow$ Relu layer (1024) $\rightarrow$ Relu layer (512) (Figure 1).

The following subsections describe the used DCNNs architectures and discuss the datasets, data augmentation techniques, and metrics used in this research.

A. Datasets

There is a large number of existing datasets to deal with the vehicle classification task [11] [12]. Some datasets are created to solve a particular case or scenario, being smaller and providing poor generalization, while other datasets are multipurpose, which have a lot of data but can bring a model performance saturation and not work for challenging scenarios [3]. This paper focused on two well-known datasets, Cars-196 [13] and CompCars [14], and an alternative dataset called BRCars [8].

Being the only dataset that does not have a validation set, Cars-196 has 16,185 images divided among 196 car models. The images were collected from Flickr, Google, and Bing. Among the datasets that already have a defined validation set, the CompCars is a dataset created with images taken from the web and surveillance camera scenes that contains 136,727 images referring to 1,716 car models.

Finally, BRCars dataset is divided into two sets: BRCars-196 and BRCars-427. For our experiments, BRCars-427 was chosen since there it has more data and unbalanced classes, containing 300,325 images referring to 427 car models. However, because the BRCars-427 contains 57,971 images at cockpit perspective of the cars, which are irrelevant to the context of this work, it was necessary to filter the data to remove such images, resulting in 242,354 useful images.

B. DCNN Architectures

For the experiments conducted in this paper, four state-of-the-art DCNNs architectures were chosen: VGG-16 [15] [12], ResNet-152 [5] [10], DenseNet-169 and Inception-V3 [3] [8].

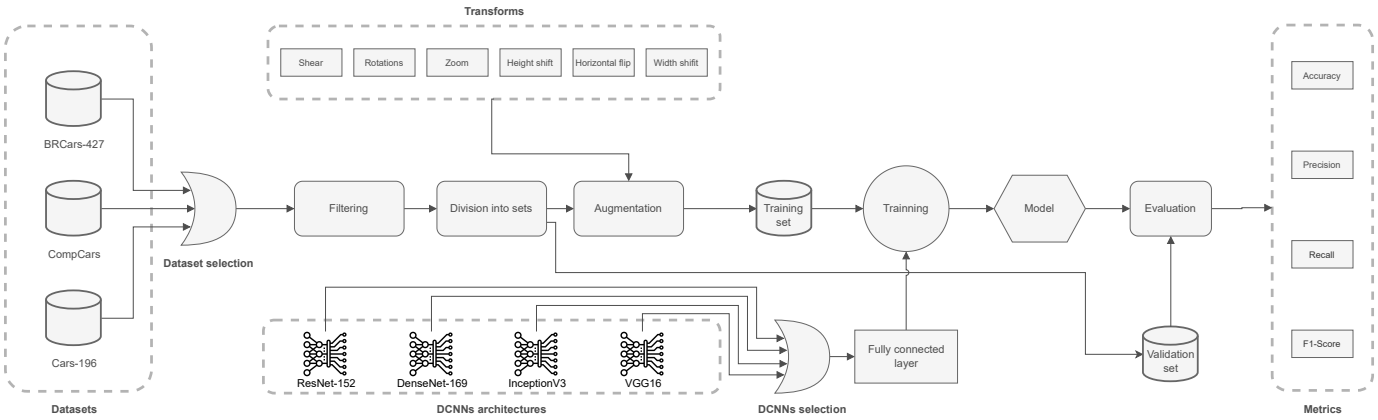

Fig. 1. Representation diagram of the experiment developed considering each step proposed in the methodology.

The VGG network has become a popular choice because of improvements in the CNNs' performance [16]. This network has 16 trainable layers and it is characterized by its simplicity, using small-size convolution filters (3x3). ResNet [17] introduced the concept of residual learning to train even deeper CNNs. It has convolutional layers that receive the input and propagate it to the output of that same layer. In this way, it is possible to train an effective deep neural network through residual blocks.

DCNNs such as Resnet, and DenseNet [18] also use the outputs concatenation. The main difference between them is the fact that the outputs in the last layer of each block are concatenated instead summed, thus, the last layer of each block is densely connected to all previous layers [19]. Many works uses variations of DenseNet, such as DenseNet-121 [12], DenseNet-161 and DenseNet-201 [10].

Inception-V3 [20] was built based on the inception blocks of the GoogleLeNet architecture, which is formed by a combination of filters with parallel paths that explore the image in a variety of filter sizes, a fact that explains the model efficiency. As in the VGG-16 architecture, the computational cost may be lower compared to ResNet and DenseNet, as they have fewer layers and parameters.

Table I presents structural details of each DCNN architecture used in our experiments.

TABLE I
DCNN ARCHITECTURES DETAILS.

Architecture	Year	N°. layers	Params
DenseNet-169	2017	169	15M
InceptionV3	2016	48	24.6M
ResNet-152	2015	152	61.2M
VGG16	2014	16	16M

C. Data Augmentation

To improve the models' performance on training, data augmentation techniques were employed, creating new transformed images from the original ones and adding it on the dataset [21] [22]. This procedure also reduces overfitting and helps to deal with unbalanced classes issues. Examples of successful implementations of data augmentation improving the results can be seen in Harianto *et al.* [23] and Nani *et al.* [24]. The strategy in this research was to apply the following transformations to increase the amount of data for each batch while maintaining the original images:

- Rescaling 1/255;
- Shear with a range of 0.15;
- Zoom transformations with a range of 0.2;
- Horizontal flip transformations with nearest fill mode;
- Height shift with a range of 0.2;
- Width shift with a range of 0.2;
- Rotations with a range of 30 degrees.

These techniques were applied due to fact that the neural networks need to get a sense of possible scenarios that may be encountered in traffic cameras, such as simulating cars in opposite ways, different view angles, and distance from the cars.

D. Evaluation Metrics

Typically, works that address the problem of fine-grained vehicle classification use accuracy as the standard evaluation metric [25]. However, accuracy can yield a superficial indication that the model can extract the desired characteristics from the cars. There are tasks within vehicle recognition that must not have a margin for false classification, such as traffic violation cars detection or criminal suspect tracking. The misclassification of cars in these contexts may lead to innocent's seizures and waste of resources chasing the wrong individuals.

Considering these scenarios and in order to reduce the number of false positives, precision, recall, f1-score and accuracy metrics were considered. Thus, using these metrics, it was possible to index the best models for the execution of the task based on an overview of the results.

IV. EXPERIMENTS

To conduct the experiments proposed in this paper, laptops with NVIDIA GeForce RTX 3060, 16GB RAM, Intel processor i-Core i7-9750H and Windows 10 operating system were

utilized. Furthermore, CUDA 11.6, Python 3.8, Keras 2.6, and Tensorflow 2.6 were the main technologies used. Once the development environment was defined, the training of each DCNN architecture-dataset combination was started from the pre-trained ImageNet weights [26]. The script used to train the models can be accessed on github [1]. It is important to note that the training was the only process running on the laptop.

The training process was performed with different setups of image resolutions (input size) and batch sizes: (i) input size of (224x244) and batch size of 16, 32, 64 and 128; (ii) input size of (144x144) and batch size of 16, 32, 64 and 128; (iii) input size of (128x128) and batch size of 32, 64 and 128.

Given the hardware limitation, the hyperparameters chosen were input size of 128x128 and batch size of 32. These values were applied to all datasets and networks to make the experiments consistent. Additionally, a learning rate of 0.00001 with Adam Optimizer was applied for 100 epochs.

Regarding the datasets, BRCars-427 and CompCars are already splitted into training, validation and testing sets [8] [14], while Cars-196 does not have a validation set. Therefore, to keep the experiments coherent, this dataset was divided into training (80%) and validation (20%), resulting in 12,948 and 3,237 images respectively.

To control the training, two callbacks were employed; early stopping to monitor the metrics and halt the training when it stops improving [27], and Reduce Learning Rate on Plateau, which changes the models' learning rate when the metrics stop improving [28]. Both callbacks consider validation loss as a metric to be monitored and have the same *delta* value of 0.0001, which means when the validation loss does not reach an improvement of at least this value, one of these callbacks will be triggered in 30 and 10 epochs, respectively.

V. Results and Discussion

The first analysis of the models derived from the architecture-dataset combination was based on the observation of evaluation metrics obtained for each of the twelve models presented in Table II. The values are organized per dataset, where the bold font represents the best performances and the italic font represents the worst performances for each metric. Given the metrics with similar values, the best performances were from ResNet-152 and DenseNet-169.

Comparing the architectures and datasets, the best metrics were found using BRCars-427. In addition, the results in CompCars were lower and this is due to the higher number of classes making the problem more complex, but still higher than 70%. As stated, the robustness of deeper architectures led to better results and this was confirmed in Cars-196. For this dataset, the results are much lower than the other two, but it was still possible to obtain regular metrics using deeper networks even with fewer data.

[1]https://github.com/souzawes/fine-grained-cars-recognition-dcnn

TABLE II
Evaluation metrics achieved in test set. **Best** and *worst* results for each dataset

Dataset	Architecture	Accuracy	Precision	Recall	F1
BRCars-427	ResNet-152	83.31	86.21	79.63	**82.05**
	DenseNet-169	**83.53**	**87.27**	**79.83**	81.90
	InceptionV3	81.18	85.22	73.05	79.05
	VGG16	*56.00*	*59.00*	*53.00*	*55.00*
Compcars	ResNet-152	74.20	81.28	72.34	70.33
	DenseNet-169	**76.30**	**84.56**	**74.60**	**72.63**
	InceptionV3	*66.16*	80.09	*64.36*	*59.56*
	VGG16	67.61	*77.00*	65.76	63.63
Cars-196	ResNet-152	55.17	67.09	55.28	50.91
	DenseNet-169	**60.41**	**74.84**	**60.31**	**52.52**
	InceptionV3	*40.62*	65.34	*39.33*	*24.39*
	VGG16	44.63	*64.96*	44.26	33.70

According to Table III, the deeper architectures (ResNet-152 and DenseNet-169) require more time to complete training and also had the longest inference times. The expressive difference in training time is directly related to the architecture's robustness. In terms of inference time, the deeper CNN also obtained worse results. However, the difference between them is very small, about 32ms as in VGG16 (faster) and DenseNet-169 (slower).

TABLE III
DCNN architectures details for this experiment using the largest dataset (BRCars-427)

Architecture	Train time	Inference time
ResNet-152	34h	75ms
DenseNet-169	28h	78ms
InceptionV3	17h	65ms
VGG16	**13h**	**46ms**

Figure 2 shows three samples from BRCars-427 dataset and the respective top 3 predicted classes, as reached in Table II values. The architectures ResNet-152, DenseNet-169 and InceptionV3 had almost 100% of confidence in the correctly predicted class, while VGG16 had the worst predictions.

Thereafter, the second analysis was the comparison of the results with the related works presented in Table IV, assuming that the best architectures in the present work are ResNet-152 and DenseNet-169. In general terms, metrics with values greater than 50% were obtained, even with hardware limitations and consequently a decrease in hyperparameter values (input size and batch size).

Compared to the networks ResNet-50 and InceptionV3 used in [8], the best architectures in this paper had slightly higher metrics on BRCars-427. They were trained for more than 75 epochs and achieved better metrics even with smaller input and batch sizes.

The same thought applies to experiments with CompCars. Metrics above 70% were obtained as in [3], despite hardware limitations. Sánchez *et al.* [3] and Ma *et al.* [5] only considered images of cars captured in the frontal view and 431 classes. In contrast, the experiments performed in this paper considered all 1,716 classes in all scenarios. This may explain the difference of more than 20% accuracy compared with Ma

TABLE IV
WORKS COMPARISON. RESNET-152 AND DESNET-169 ACHIEVED THE BEST RESULTS.

Dataset	Architecture	Epochs	Input size	Batch size	Accuracy	Precision	Recall	F1
BRCars-427	**ResNet-152 (present)**	100	128x128	32	83.31	86.21	79.63	**82.05**
	DenseNet-169 (present)	100	128x128	32	**83.53**	**87.27**	**79.83**	81.90
	InceptionV3 [8]	25	256x256	64	-	82.00	79.00	79.00
	Resnet-250 [8]	25	256x256	64	-	80.00	77.00	79.00
Compcars	**ResNet-152 (present)**	100	128x128	32	74.20	81.28	72.34	70.33
	DenseNet-169 (present)	100	128x128	32	76.30	84.56	74.60	72.63
	DenseNet161-CMP [5]	300	224x224	32	**97.89**	-	-	-
	ResNet152-CMP [5]	300	224x224	32	97.01	-	-	-
	VGG16-CMP [5]	300	224x224	32	97.80	-	-	-
	InceptionV3 [3]	50	-	-	78.47	-	-	-
Cars-196	**ResNet-152 (present)**	100	128x128	32	55.17	67.09	55.28	50.91
	DenseNet-169 (present)	100	128x128	32	60.41	**74.84**	**60.31**	**52.52**
	VGG16 [9]	100	224x224	32	89.67	-	-	-
	DenseNet-161 [9]	100	224x224	32	93.37	-	-	-
	ResNet152 [10]	100	-	32	**93.96**	-	-	-
	DenseNet201 [10]	100	-	32	92.81	-	-	-

Fig. 2. Top 3 predicted classes for sample images from test set of BRCars-427 using all trained models. The correct predictions are in green, while the incorrect ones are in red.

et al. work, since in the images used the cars do not vary in angle and position. In addition, they trained the networks for 300 epochs.

The analysis of metrics using Cars-196 was similar to using CompCars. The authors obtained accuracies greater than 90%. As mentioned in section III-A, Cars-196 does not have a validation set and this could cause poor generalization and overfitting. For this paper, the Cars-196 training set was split 80/20 for training and validation, as specified in Section IV. This could be reflected in low metrics as the entire dataset has few images. Another fact that may have led to high accuracies in Hassan *et al.* work [10] is the usage of high-performance machines.

VI. CONCLUSION

Technology development in the context of Intelligent Transportation Systems is becoming increasingly important today. Specifically, advances in computer vision, distributed computing, cloud computing and embedded systems have enabled the deployment of devices capable of extracting information from images captured in various locations throughout a city. One of the most common ITS tasks is the vehicle characteristics identification, known as fine-grained vehicle classification, which can be applied in contexts ranging from parking management systems to urban safety applications such as tracking criminals' vehicles.

This study presented a comparison of performance among multiples combinations of architectures-datasets to determine which combination works better for the fine-grained vehicle classification problem for car make and model recognition. Based on the results presented throughout this paper, it was possible to conclude that the deeper CNN architectures (ResNet-152 and DenseNet-169) are the most recommended for the task, as they achieved the best test results.

The experiments conducted in this paper also revealed that among the used datasets the BRCars-427 produced the most accurate models and produced good results even for the InceptionV3, making it a lightweight option for implementing

the car classifier based on make and model. On the other hand, the dataset Cars-196 presented the worst result mainly due to its small amount of training and validation data, which does not allow a model generalization and caused very low metrics in the test set.

Based on the results obtained in this paper, it is possible to conclude that ResNet-152 and DenseNet-169 are the most suitable architectures among those tested for the fine-grained vehicle classification task. For future works, it is planned to analyze techniques that can improve training for small datasets such as Cars-196, as well as the coverage of more Convolutional Neural Networks such as other variants of ResNets and DenseNets, Inception-v4 and EfficientNet.

ACKNOWLEDGEMENT

The results presented in this paper have been developed as part of a project between SiDi, financed by Samsung Eletrônica da Amazonia Ltda., under the auspices of the Brazilian Federal Law of Informatics no. 8248/91.

REFERENCES

[1] A. B. D. S. PÚBLICA, "Anuário brasileiro de segurança pública," 2020.

[2] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1840–1852, 2020.

[3] H. C. Sánchez, N. H. Parra, I. P. Alonso, E. Nebot, and D. Fernández-Llorca, "Are we ready for accurate and unbiased fine-grained vehicle classification in realistic environments?" *IEEE Access*, vol. 9, pp. 116 338–116 355, 2021.

[4] K. Valev, A. Schumann, L. Sommer, and J. Beyerer, "A systematic evaluation of recent deep learning architectures for fine-grained vehicle classification," in *Pattern Recognition and Tracking XXIX*, vol. 10649. International Society for Optics and Photonics, 2018, p. 1064902.

[5] Z. Ma, D. Chang, J. Xie, Y. Ding, S. Wen, X. Li, Z. Si, and J. Guo, "Fine-grained vehicle classification with channel max pooling modified cnns," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3224–3233, 2019.

[6] Y. Ding, Z. Ma, S. Wen, J. Xie, D. Chang, Z. Si, M. Wu, and H. Ling, "Ap-cnn: weakly supervised attention pyramid convolutional neural network for fine-grained visual classification," *IEEE Transactions on Image Processing*, vol. 30, pp. 2826–2836, 2021.

[7] S. Aneeba Najeeb, R. Hammad Raza, A. Yusuf, and Z. Sultan, "Fine-grained vehicle classification in urban traffic scenes using deep learning," *arXiv e-prints*, pp. arXiv–2111, 2021.

[8] D. M. Kuhn and V. P. Moreira, "Brcars: a dataset for fine-grained classification of car images," in *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2021, pp. 231–238.

[9] X. Li, L. Yu, D. Chang, Z. Ma, and J. Cao, "Dual cross-entropy loss for small-sample fine-grained vehicle classification," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4204–4212, 2019.

[10] A. Hassan, M. Ali, N. M. Durrani, and M. A. Tahir, "An empirical analysis of deep learning architectures for vehicle make and model recognition," *IEEE Access*, vol. 9, pp. 91 487–91 499, 2021.

[11] F. Tafazzoli, H. Frigui, and K. Nishiyama, "A large and diverse dataset for improved vehicle make and model recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 1–8.

[12] L. Lu, P. Wang, and H. Huang, "A large-scale frontal vehicle image dataset for fine-grained vehicle categorization," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[13] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *Proceedings of the IEEE international conference on computer vision workshops*, 2013, pp. 554–561.

[14] L. Yang, P. Luo, C. Change Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3973–3981.

[15] C. N. Benavides and C. Tae, "Fine grained image classification for vehicle make and model using convolutional neural network," 2019.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[19] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[21] M. R. Aquino, M. Gutoski, L. Hattori, and H. S. Lopes, "The effect of data augmentation on the performance of convolutional neural networks," in *Brazilian Society of Computational Intelligence*, 2017.

[22] J. Nalepa, M. Marcinkiewicz, and M. Kawulok, "Data augmentation for brain-tumor segmentation: A review," in *Frontiers in Computational Neuroscience*, 2019.

[23] R. A. Harianto, Y. M. Pranoto, and T. P. Gunawan, "Data augmentation and faster rcnn improve vehicle detection and recognition," in *3rd East Indonesia Conference on Computer and Information Technology*. EIConCIT, 2021, pp. 128–133.

[24] L. Nanni, M. Paci, S. Brahnam, and A. Lumini, "Comparison of different image data augmentation approaches," in *Journal of Imaging*, 2021.

[25] M. Buzzelli and L. Segantin, "Revisiting the compcars dataset for hierarchical car classification: New annotations, experiments, and results," *Sensors*, vol. 21, no. 2, p. 596, 2021.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[27] B. J. Kim, "Improved deep learning algorithm," *Journal of Advanced Information Technology and Convergence*, vol. 8, no. 2, pp. 119–127, 2018.

[28] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 1100612.