ISSN 1870-4069

Vehicle Make and Model Recognition with Generation of New Classes Using Clustering Techniques

Diana Itzel Vázquez-Santiago, Héctor Gabriel Acosta-Mesa, Efrén Mezura-Montes

Universidad Veracruzana, Artificial Intelligence Research Institute, Mexico

zs21000454@estudiantes.uv.mx, {heacosta, emezura}@uv.mx

Abstract. One of the main problems faced by supervised learning classification algorithms is scalability. No matter how good their classification accuracy is, they are not able to classify objects for which they were not trained. In this paper we propose a solution to this problem specifically aimed at vehicle make and model recognition. We used a Convolutional Neural Network (CNN) for classification and feature extraction, addressing the scalability problem by using two clustering techniques: K-means and MOCK. For the generation of new classes, we used the feature vectors extracted by the CNN of the images that do not belong to any of the classes with which the model was trained. The results showed that with the learning generated by a CNN it is possible to generate feature vectors with similarities for objects of the same class even if the network was not trained to classify them, which made it possible to generate new classes using unsupervised learning such as clustering.

Keywords: Scalability, CNN, clustering, MOCK.

1 Introduction

Automatic vehicle makes and model recognition aims to offer innovative services to improve the efficiency and safety of transportation networks. Some of these services are intelligent traffic analysis and management, electronic toll collection, emergency vehicle notifications, automatic enforcement of traffic rules, etc. The main problem, in the specific case of this application domain, is that most of the applied approaches for vehicle make and model recognition require large amounts of data to correctly train a model.

It is estimated that there are currently more than 3,300 vehicle makes in the world, which have added and removed models from the market, modifying the design in each generation and producing different versions of the same vehicle which has made impossible to have a database containing all existing vehicles.

Algorithm 1: PESA-II Pseudocode

```
1   Initialize a random (internal) population IP
2   Evaluate each member of IP
3   Initialize the external population EP to the empty set
4   repeat
5   Incorporate non-dominated vectors from IP into EP
6   Delete the current contents of IP
7   repeat
8   With probability Pc, select two parents from EP, where Pc= Crossover
    probability
9   Produce a single child via crossover
10  Mutate the child created in the previous step
11  With probability 1 – Pc, select one parent
12  Mutate the selected parent to produce a child
13  until the population IP is filled;
14  until termination criteria is met;
15  Return the members of EP as the result
```

This limitation leaves us with a scalability problem that results in vehicles that cannot be classified correctly because the algorithms were not trained to recognize them.

In this work, we propose a feasibly solution to the scalability problem of classification algorithms that use supervised learning by using clustering algorithms to generate new classes using the feature vectors extracted by our classification algorithm. In the state of the art, the scalability problem has been addressed by authors such as Nazemi et al. [1] from an anomaly detection approach.

Their base system is capable of classify 50 specific vehicle models, to which they added an anomaly detection to identify vehicles that do not belong to any of the 50 classes, to subsequently classify them based on their dimensions within 2 new classes: "Unknown heavy" and "Unknown light".

Other authors such as Kezebou et al. [4] proposed a Few-Shots Learning approach requiring between 1 and 20 images for the generation of new classes.

2 Methodology

For this work, the VMMRdb database [2] was used since it is one of the most cited in the specialized literature. With the intention of simplifying the problem for analysis, only five classes were used: Dodge Grand Caravan 2005, Ford Explorer 2002, Ford Mustang 2000, Nissan Altima 2005 and Toyota Camry 2007 to train a CNN whose architecture was proposed in the Microsoft technical documentation library [3] with which training and testing times of 2m24s were achieved with accuracies between 90%-95% in 5 epochs.

Since the main objective of this project is to have an algorithm capable of classifying vehicles even if they do not belong to any predefined class, an algorithm was designed

Table 1. Comparison between data volumes per class.

Class	Images (Rear view)	Training (Before augmentation)	Training (After augmentation)	Test (No augmentation)
Dodge Grand Caravan 2005	164	144	1,000	20
Ford Explorer 2002	234	214	1,000	20
Ford Mustang 2000	216	196	1,000	20
Nissan Altima 2005	294	274	1,000	20
Toyota Camry 2007	170	150	1,000	20

to generate new classes from the clustering of images that do not belong to the classes with which the CNN was trained.

The feature vectors of the images (extracted with the CNN) were used to perform clustering using two algorithms for comparison purposes. The first clustering algorithm is K-means.

The original version of the algorithm was implemented in Python and in the interests of have an additional comparison, a second version was developed using the sklearn library, which implements the K-means clustering algorithm. See section 2.3 for more details.

The second clustering algorithm named MOCK employs a multi-objective evolutionary approach named PESA-II. This algorithm attempts to minimize two objectives that are in conflicting with each other (intra-cluster variation and the number of clusters).

The concept of Pareto dominance is used to find a set of different non-dominated clustering solutions that achieve a good trade-off between the two objectives. PESA-II's pseudocode is shown in Algorithm 1. See section 2.4 for more details.

2.1 Image Preprocessing

The images went through a few processes to fit the model. The first was to segment the images of each class according to the views they showed (front, rear, and side). Due to the scope of the project, we work only with the rear views of the five classes mentioned in Section 2.

Table 1 shows the volume of images (rear views) available per class. Twenty images from each class were chosen for testing and the rest for training, however, as can be seen in Table 1 the volume of images was low and not balanced to adequately train the CNN.

To solve these problems, data augmentation and balancing processes were performed on the training set. After these processes, the final number of images for training per class was 1,000 and the twenty images that were originally selected for testing were kept without data enhancement.

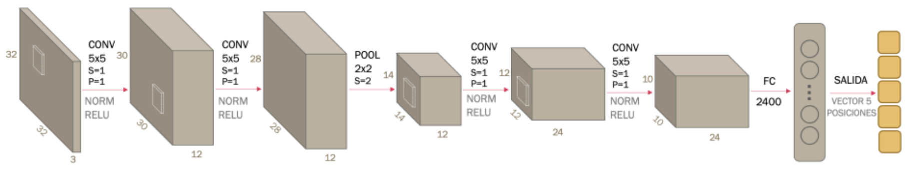

Fig. 1. Architecture of the convolutional neural network used and proposed in the Microsoft technical documentation library [3].

The next process was to reduce the dimensions of the images (training and testing) since the CNN architecture required input dimensions of 32x32x3. With the processes mentioned above, the classification model achieved accuracies between 80%-85%.

To improve recognition accuracy the images were cropped to preserve only the region of interest (ROI), which improve the accuracy by up to 10%.

2.2 Convolutional Neural Network (CNN)

This project was implemented in Python mainly because of the access to the PyTorch library which provides support for the development of applications related to machine learning, computer vision, natural language processing, etc. More specifically, it has a base class for all Convolutional Neural Networks modules, which facilitates its implementation and execution.

Initially in this project, it was proposed to work with well-known CNN architectures such as AlexNet or VGG, however, in the first stages, tests were carried out and execution times were time consuming (30min-50min) achieving a maximum accuracy of 60%.

Because of this, we chose to use an architecture found in the Microsoft technical documentation library [3], which can be seen in Fig 1. Originally, the network was designed to work with the CIFAR10 database, so the input dimensions were 32x32x3.

Even with the possible loss of information, it was decided to keep this architecture and resize the images of the VMMRdb database to fit, since even without the data balance, the cropping of the ROI and with the resizing, accuracies of 70% were achieved. The only modification to the architecture was to change the output of the fully connected layer to five (number of classes).

In the training stage, 20-image subsets of the training data were generated and reorganized at each epoch to reduce overfitting. For the update of the network weights during training, the Adam optimizer was used with a learning rate of 0.001 and a weight decrease of 0.0001.

Finally, for the performance evaluation, two indicators were used in each epoch, the first indicator was to evaluate the classification accuracy of the network with the whole test set (20 images as shown in Table 1) and the second indicator was with the Cross Entropy loss function which is mathematically expressed in (1). Where i is the class, c is the number of classes, y_i is the actual class and $\hat{y}_i$ is the predicted class:

$$-\sum_{i=1}^{c} y_i \log \hat{y}_i, \tag{1}$$

Table 2. Comparison between Convolutional Neural Network executions. The execution with the most accurate result is highlighted in bold.

Execution	Cross Entropy Loss	Accuracy	Accuracy per class				
			Dodge Grand Caravan 2005	Ford Explorer 2002	Ford Mustang 2000	Nissan Altima 2005	Toyota Camry 2007
1	0.294	**95%**	95%	100%	95%	90%	95%
2	0.290	91%	100%	85%	90%	90%	90%
3	0.295	92%	100%	95%	90%	95%	80%
4	0.313	93%	95%	95%	85%	95%	95%
5	0.300	91%	95%	100%	95%	90%	75%
6	0.311	92%	90%	100%	85%	100%	85%
7	0.247	90%	100%	95%	90%	90%	75%
8	0.313	93%	100%	90%	90%	85%	100%
9	0.303	91%	90%	85%	95%	95%	90%
10	0.321	94%	100%	85%	95%	95%	95%
Average		92.2%	97%	93%	91%	93%	88%

2.3 K-means Clustering

As mentioned above, the aim of this project is to have an algorithm capable of classifying vehicles even if they do not belong to any predefined class. The first approach implemented for the generation of new classes was K-means clustering. Two implementations of this algorithm were developed.

The first was implementing the classical version of the algorithm. The second was using the version developed by sklearn library, initializing the centroids of the clusters with the Lloyd algorithm.

We decided to test the k-means algorithm in its two versions with nine images of three unknown classes to the CNN (three images of each class), to confirm if the clustering algorithms were able to group the images by model. The clustering process was performed using the feature vectors extracted by the CNN, which, as can be seen in Fig 1. had 2,400 features.

2.4 MOCK Clustering with Multi-Objective Evolutionary Approach (PESA-II)

The second approach implemented for the generation of new classes was a multi-objective clustering algorithm with automatic determination of the number of clusters (MOCK) optimized with a multi-objective evolutionary algorithm (MOEA), called PESA-II proposed by Corne et al. [5].

The encoding of individuals uses a representation where each individual *g* is made up of *N* genes, *g1,...,gN,* where *N* is the number of data to be clustered and the value *j*

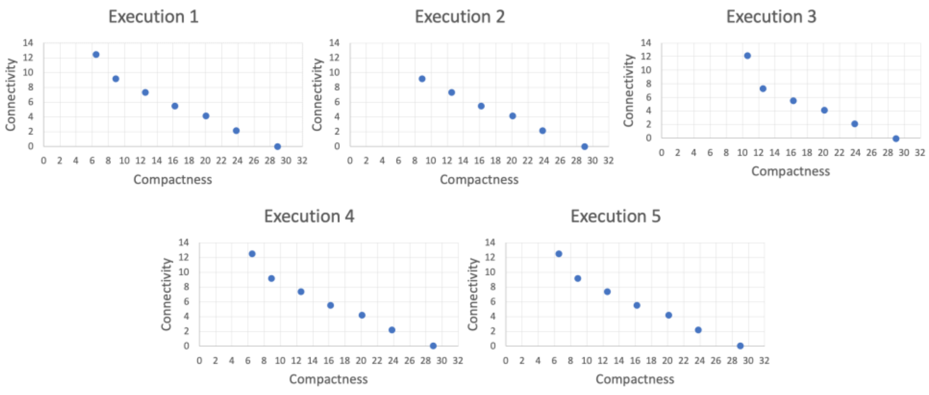

Fig. 2. Pareto fronts obtained in 5 executions of the MOCK algorithm.

assigned to the *i*-th gene represents a union between the *j* and *i* data. A minimum spanning tree (MST) was generated with the Prim algorithm to initialize the population.

The MST represented the first individual and for the subsequent generation of the population the (i - 1) longest connections were eliminated. In the genotype, this was reflected by assigning to itself the gene representing the connection. The decoding of this representation requires the identification of all the subgraphs since the data belonging to each subgraph is assigned to a cluster.

The variation operators used are the uniform crossover and the nearest neighbor mutation operator, which limits the search space since it can only generate connections between the nearest neighbors of the gene being mutated. In this approach, there are two objective functions to be minimized:

1. **Cluster Compactness** or global deviation, which is calculated by summing the distances between each datum and its corresponding centroid in a given cluster and is mathematically represented as:

$$\mathrm{Dev}(C) = \sum_{C_k \epsilon C} \sum_{i \epsilon C_k} \delta(i, \mu_k), \tag{2}$$

where C is the set of clusters M_k is the centroid of cluster C_k, i is each element of the data set and δ(...) is the Euclidean distance.

2. **Cluster Connectivity** which evaluates if the nearest neighbors of an element have been placed in the same cluster as the current element and is mathematically represented as:

$$\mathrm{Conn}(C) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} x_{i,nn_i(j)} \right), \text{ donde } x_{r,s} = \begin{cases} \frac{1}{j} \; si \nexists C_k : r, s \in C_k, \\ 0 \text{ de lo contrario,} \end{cases} \tag{3}$$

where C is the set of clusters, N is the amount of data in the dataset, L is the amount of nearest neighbors (user-defined parameter), $nn_i(j)$ is the jth nearest neighbor and $x_{r,s}$ is the penalty function.

There will only be penalties if any *j*th nearest neighbor is not in the same cluster as the *i*th data. PESA- II's pseudocode is shown in Algorithm 1 where the use of two populations of solutions can be highlighted: IP which has a fixed size and is responsible for exploring new solutions and EP which has a limited, but not fixed size and has the job of exploiting good solutions since it consists of "niches" distributed over the objective space (Pareto Front).

In each generation, the generated solutions (IP) are evaluated and those that are in the objective space are selected to become part of EP, preferring those solutions that occupy less crowded spaces (the "niches" are avoided) to try to completely cover the Pareto Front.

3 Results

To address the problem of scalability of classification algorithms that use supervised learning, it is essential for our proposal to have a classifier with good recognition accuracy to differentiate between images that belong to the predefined classes from those that do not.

To test the classification accuracy of our model the CNN implemented and detailed in Section 2.2 was trained and tested 10 times with the 5 classes of the VMMRdb database [2] mentioned in Section 2 which went under the data augmentation and balancing processes mentioned in Section 2.1. Table 2 shows the accuracy results obtained in the 10 training-testing executions.

Due to the scope of the project, it is left as future work the implementation of a novelty detection technique to automatically detect images that do not belong to the predefined classes to which the clustering will be applied to generate new classes.

Given the above, in order to show that the K-means and MOCK algorithms were able to cluster the images of the unknown classes for the CNN and thus generate new classes, nine images of three unknown models (three of each class) were entered into the CNN: Ford Ranger 2019, Toyota Prius 2019 and Volkswagen Beetle 2013 which, as expected, generated a misclassification as it was not trained for those classes, however, what was important in this case was to obtain the feature vectors generated by the CNN to enter them into the clustering algorithms.

For the execution of MOCK the parameters were calibrated as follows: Number of generations = 100, Maximum External Population Size = 15, Internal Population Size = 8, L nearest neighbors = 3, Crossover Probability = 0.5. Fig. 2 shows the Pareto Fronts obtained in five executions of the algorithm.

It is important to remember that the clustering process by definition is subjective. Also, it should be considered that each time the CNN is trained, it will learn in a unique way and will be reflected in the weights that are set at the end of the training, therefore, the features extracted after each training will depend on those learned weights.

Taking these two points into consideration, five tests of the clustering algorithms were performed using five sets of feature vectors obtained from five executions of the CNN with different weights. Only with one of the sets the desired clustering was achieved by both clustering approaches as shown in Fig. 3.

Initially the hypothesis was that MOCK would outperform K-means, however the results showed similar performances where the only advantage shown by MOCK was the automatic determination of the number of clusters. To get a better understanding of the results, five distance matrices of the five sets of feature vectors extracted by the

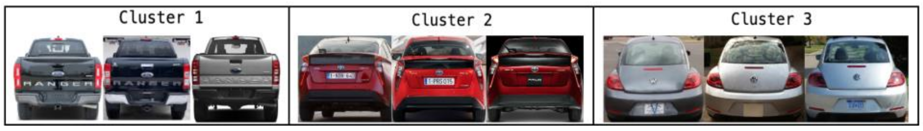

Fig. 3. Classification expected and obtained with the K-means and MOCK algorithms.

CNN were performed using the Euclidean distance since the same metric was used in the clustering algorithms.

This test showed that only in one of the matrices the feature vectors belonging to the same classes were spatially close and it was with that set that the clustering algorithms achieved the desired clustering.

One point to highlight with the K-means approach is that the addition of the Lloyd algorithm implemented by sklearn library for the initialization of the centroids gave it a great advantage over the classical version that did not achieve the expected classification in any of the executions.

Finally, it was noted that in most cases the clustering were related to the predominant shades in the images, which indicates that it may be preferable to work with grayscale images to prevent bias.

4 Conclusions and Future Work

In this work a feasibly solution to the scalability problem was presented, confirming that it is possible to generate new classes using clustering algorithms to group images based on the feature vectors extracted by the classification model even if the classes are unknown. However, it was observed that the feature vectors belonging to the same classes were not in close regions in terms of Euclidean distance.

The research carried out after this project reflected that this distance metric is recommended only to compare points in two or three dimensions. In larger dimensional spaces, all points tend to be far apart, then other measures will be explored such as the cosine distance. Another point to consider is that the feature vectors will depend on the training of the network, specifically the learned weights, which can influence negatively.

To achieve a better control, we propose the use of neuroevolution of CNNs with an objective function that measures the consistency of the feature maps. By doing this, we could get feature vectors located spatially close if they belong to the same class, and far away if they belong to different classes. In the literature review, this has been achieved using techniques such as Contrastive Loss [7].

Regarding the clustering algorithms, according to authors such as Martínez-Peñaloza et al. [6], better results were achieved using the MOEA NSGA-II compared to the original version of MOCK, which uses PESA-II.

Since in this work there were no explicit differences between K-means and MOCK, it is proposed to use the optimized version with NSGA-II to try to achieve better results. Finally, it is proposed as future work to increase the number of classes and perform a novelty detection to automatically recognize vehicles that do not belong to the labels with which the CNN is trained and perform clustering on them.

Acknowledgments. The authors would like to thank the Consejo Nacional de Ciencia y Tecnología (CONACYT), an institution of the Government of Mexico, for the financial support provided through the "Beca Nacional" as part of the Programa de Becas para Estudios de Posgrado.

References

1. Nazemi, A., Azimifar, Z., Shafiee, M., Wong, A.: Real-time vehicle make and model recognition using unsupervised feature learning. IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 7, pp. 3080–3090 (2019). DOI: 10.1109/TITS.20 19.2924830.
2. Tafazzoli, F., Frigui, H., Nishiyama, K.: A large and diverse dataset for improved vehicle make and model. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1–8 (2017). DOI: 10.1109/CVPRW.2017.121.
3. Kezebou, L., Oludare, V., Panetta, K., Agaian, S.: Few-shots learning for fine-grained vehicle model recognition. In: IEEE International Symposium on Technologies for Homeland Security, pp. 1–9 (2021). DOI: 10.1109/HST53381.2021.9619823.
4. Corne, D. W., Jerram, N. R., Knowles, J. D., Oates, M. J.: PESA-II: region-based selection in evolutionary multiobjective optimization. In: Proceedings of the 3rd annual conference on genetic and evolutionary computation, pp 283–290 (2001). DOI: 10.5555/29552 39.2955289.
5. Martínez-Peñaloza, M. G., Mezura-Montes, E., Cruz-Ramírez, N., Acosta-Mesa, H. G., Ríos-Figueroa, H. V.: Improved multi-objective clustering with automatic determination of the number of clusters. Neural Computing and Applications, vol. 28, no. 8, pp. 2255–2275 (2017). DOI: 10.1007/s00521-016-2191-1.
6. Wang, F., Liu, H.: Understanding the behaviour of contrastive loss. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2495–2504 (2021). DOI: 10.1109/CVPR46437.2021.00252.