A 2D Deep Boltzmann Machine for Robust and Fast Vehicle Classification

Daniel Felipe Silva Santos
UNESP - São Paulo State University
Bauru, São Paulo, Brazil
danielfssantos1@gmail.com

Gustavo Botelho de Souza
UFSCar - Federal University of São Carlos
São Carlos, São Paulo, Brazil
gustavo.botelho@gmail.com

Aparecido Nilceu Marana
UNESP - São Paulo State University
Bauru, São Paulo, Brazil
nilceu@fc.unesp.br

***Abstract*—The visual and automatic classification of vehicles plays an important role in the Transport Area. Besides of security issues, the monitoring of the type of traffic in streets and highways, as well the traffic dynamics over time, allows the optimization of use and of resources related to such public infrastructure. In this work we propose a novel method, called 2D-DBM, for robust and efficient automatic vehicle classification through color images based on a DBM (Deep Boltzmann Machine) combined with bilinear projections. While the DBM training allows a robust initialization of discriminative MLP (Multilayer Perceptron) neural network parameters, the bilinear projection technique can scale down the MLP dimensions, obtaining efficiency while preserving accuracy. The proposed method was assessed on the BIT-Vehicle database, a challenging dataset consisting of frontal images of vehicles collected in a real traffic environment, and compared with a CNN (Convolutional Neural Network) and a traditional DBM (without bilinear projection). The obtained results show that, while keeping the accuracy, the new method significantly reduced the network size and the processing time.**

***Keywords*-Vehicle Classification; Traffic Control; Image Analysis; Deep Boltzmann Machines; Bilinear Projection.**

I. Introduction

Intelligent Transportation Systems (ITS) have gained considerable attention in the past few years due to their valuable applications, such as avoidance of vehicle collision [1], [2] and traffic flow monitoring [3], [4], [5].

An ITS, in general, is composed of different modules, such as vehicle detection and vehicle tracking, that communicate with each other [3]. The vehicle classification module, usually placed at the top of the ITS stack, is responsible for taking important decisions concerning the entire information already processed by the other components, e.g., classificating the detected vehicle in the images into predefined classes.

In this work, we propose a robust and efficient method for automatic visual vehicle classification based on a novel deep learning architecture, inspired by works such as [6], [7], [8], which achieved good results in such task. According to [9], the main advantages of using deep learning architectures consist in the fact that: (i) they present an intrinsic contrast detection ability (acting similar to edge detection frameworks); (ii) they work with more robust (high-level) features than traditional methods, learned automatically from training data even by mean of unsupervised algorithms; and (iii) their learned parameters can also serve as a good initialization for other models, such as a Multilayer Perceptron (MLP) network - as proposed in section III.

Despite of such advantages, there are two main problems associated with the already existent deep learning-based techniques. The first issue concerns to the amount of time and computer processing demanded to deal with their huge amount of matrix products (due to their multilayer architectures) as well as with their sampling and second order optimization methods, like conjugate gradient [10], [11]. The second problem regards to the choice of the best network configuration, i.e., determining its optimal number of layers and amount of neurons in each of them.

In this work we also address such issues, especially the time consumption and network size definition. In order to solve the time consumption problem, especially for training the network, parallel programming in a CUDA (Compute Unified Device Architecture) device and a recently proposed deep learning architecture, the Deep Boltzmann Machine (DBM) [12], are employed. The problem of defining the amount of neurons in the network is addressed by using a bilinear projection technique [13], which also improves the network efficiency, reducing its dimensions while preserving accuracy.

II. Technical background

In this section, fundamental concepts concerning the main techniques used in the proposed deep network for visual vehicle classification are briefly described. They involve, mainly, fundamentals of Boltzmann Machines (BM) [14], Deep Boltzmann Machines (DBM) [12] and 2D-Linear Discriminant Analysis [13].

A. Boltzmann and Deep Boltzmann Machines

Boltzmann Machines (BM) [14] are neural networks that can also be viewed as fully connected graphs, in which all nodes are connected to each other by symmetrical edges. These nodes are treated as stochastic binary variables and can be virtually grouped in two layers, $\boldsymbol{v}$ and $\boldsymbol{h}$, as shown in Fig. 1, in order to facilitate the formulation of the model. There is an important property directly related to Energy Models [14] which states that, when the BM reaches the thermal equilibrium, its joint probability can be written as:

$$P(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\Theta}) = \frac{e^{-E(\boldsymbol{v}, \boldsymbol{h})}}{Z(\boldsymbol{\Theta})}, \tag{1}$$

in which $\boldsymbol{v}$ and $\boldsymbol{h}$ denote the visible and hidden layers of the BM, respectively, $\boldsymbol{\Theta} = \{\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{W}\}$ stands for the BM set of parameters, and the normalization term $Z(\boldsymbol{\Theta})$, also known as the partition function, is given by:

$$Z(\boldsymbol{\Theta}) = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}. \tag{2}$$

The Energy of the model joint distribution is given by:

$$E(\boldsymbol{v},\boldsymbol{h}) = -\sum_{i<j} v_i v_j \boldsymbol{L}_{ij} - \sum_{k<l} h_k h_l \boldsymbol{S}_{kl} - \sum_{i,k} v_i h_k \boldsymbol{W}_{ik}. \tag{3}$$

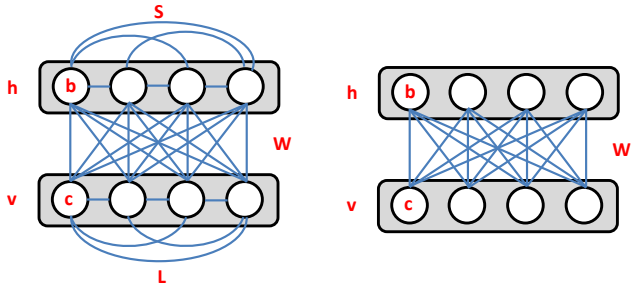
in which j and l indicates respectively the maximum quantity of nodes in layer $\boldsymbol{v}$ and in layer $\boldsymbol{h}$.

In an attempt to solve the optimization (learning) problem of the BM, that consists in minimizing the energy $E(\boldsymbol{v},\boldsymbol{h})$ of Eq. 3, the same as maximizing the "Goodness", $-E(\boldsymbol{v},\boldsymbol{h})$, Hinton and Sejnowsky [14] proposed a parallel version for the Simulated Annealing algorithm created by Kirkpatrick, Gelatt and Vecchi [15], capable to find the minimum of the non-linear energy function. Simulated Annealing technique tries to find the best possible minimum through the calculation of the probability ratio between the configurations of the network nodes following:

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T}, \tag{4}$$

in which α and β represent two global configurations of the network, and the variable T controls the temperature of the system (which tends to decay over time). High temperatures avoid the algorithm getting stuck in poor local optimal positions in search space, while low temperatures prevent the algorithm escaping from a great optimal solution.

Trying to overcome the convergence instability of the Simulated Annealing method, in [16] and [17] a better approach to solve the energy minimization problem is proposed based on the simplification of the BM model, which is converted to a bipartite graph structure called Restricted Boltzmann Machine (RBM), also shown in Fig. 1.



(a) (b)

Fig. 1. Boltzmann architectures: (a) Boltzmann Machine (BM); and (b) Restricted Boltzmann Machine (RBM). $\boldsymbol{L}$, $\boldsymbol{S}$ and $\boldsymbol{W}$ indicate the weights of connections between neurons, and $\boldsymbol{b}$ and $\boldsymbol{c}$ their biases. In RBMs no intralayer connections are allowed.

The optimization method presented in [16] and [17] was called Contrastive Divergence (CD) and its main goal is to solve the maximum log-likelihood approximation problem:

$$\frac{\partial\ log\ P(\boldsymbol{v})}{\partial W_{ik}} = \left\langle v_i h_k \right\rangle_{data} - \left\langle v_i h_k \right\rangle_{model}, \tag{5}$$

which consists in minimizing the divergences between two correlations, one with respect to the observed data, expectation $\langle v_i h_k \rangle_{data}$, and the other with respect to the model non-observed data, expectation $\langle v_i h_k \rangle_{model}$. Since it is difficult to calculate the model expectations, in [16] a Gibbs sampling technique is applied to the RBM until the Markov Chain hits the thermal equilibrium. In [17], the authors show that if the Markov Chain is initialized on the observed data, a single Gibbs sampling iteration is sufficient to produce the model expectations.

The conditional probabilities used in the CD method are given by the logistic functions:

$$P(h_k = 1|\boldsymbol{v}) = \frac{1}{1 + e^{(-b_k - \sum_i v_i w_{ik})}}, \tag{6}$$

and,

$$P(v_i = 1|\boldsymbol{h}) = \frac{1}{1 + e^{(-c_i - \sum_k h_k w_{ik})}}, \tag{7}$$

and, in case the RBM input values are not sparse enough (real-valued data), Eq. 7 must be replaced by the normal distribution over the training data given by:

$$P(v_i|\boldsymbol{h}) = \mathcal{N}\left(v_i \middle| \sum_k h_k w_{ik} + c_i, \sigma_i^2\right). \tag{8}$$

A Deep Boltzmann Machine (DBM) model [12], which architecture is shown in Fig. 2, is a deep model that, besides incorporating the advantages of the Deep Belief Networks (DBN) [9], such as the ability to learn internal representations from the input data (higher layers capture complex statistical structures) and a fast way of making inferences (hidden layers state computation), can also incorporate top-down feedback, making possible to use higher level knowledge to solve uncertainty regarding raw or intermediate level features with accuracy. Its optimization, as given in [9], consists in finding the solution to the maximization problem of the variational lower bound on the log-likelihood:

$$log\ P(\boldsymbol{v};\boldsymbol{\Theta}) \geq \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu})\ log\ P(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\Theta}) + \mathcal{H}(Q), \tag{9}$$

where $\mathcal{H}(\cdot)$ indicates the entropy functional.

For simplicity and speed, in [12] the authors use the fully factorized Mean-Field approximation method [18], approximating the true posterior $P(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\Theta})$ by:

$$Q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu}) = \prod_{l=1}^{L} \prod_{k=1}^{\mathcal{F}_l} q(h_k^l), \tag{10}$$

where $q(h_k^l = 1) = \mu_k^l$, $\mathcal{F}_l$ is the number of hidden units in hidden layer l, and L is the number of hidden layers.

The Mean-Field approximation is used to calculate the data expectations (middle term of Eq. 5) for the whole DBM, meanwhile due to the difficult in calculating the model expectations (last term of Eq. 5), a Persistent Contrastive Divergence (PCD) technique [19] is applied to a set of uniformly random initialized fantasy particles. After computed the data and model expectations, the final step of the stochastic approximation process consists in calculating the approximate log-likelihood gradient as in Eq. 5.

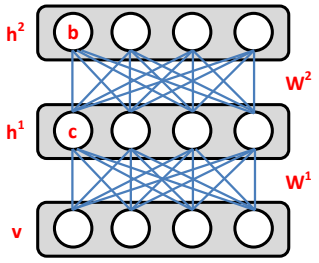


Fig. 2. DBM architecture with two layers, $\mathbf{h^1}$ and $\mathbf{h^2}$. The DBM can be viewed as a stack of RBMs with an improved learning.

B. 2D-Linear Discriminant Analysis

Given a set of t matrices $\boldsymbol{A} = \{\boldsymbol{X}^1, \boldsymbol{X}^2, \boldsymbol{X}^3, \ldots, \boldsymbol{X}^t\}$, with $\boldsymbol{X}^s \in \mathbb{R}^{m \times n}$, $\bar{\boldsymbol{A}}$ denoting the mean matrix of $\boldsymbol{A}$ and $\bar{\boldsymbol{A}}^i$ denoting the mean of all matrices from $\boldsymbol{A}$ which belongs to class i, an special kind of bilinear projection, the 2D-Linear Discriminant Analysis (2D-LDA), aims to produce a compact representation for matrix $\boldsymbol{X}^s$, denoted here by $\boldsymbol{D}^s$, such that $\boldsymbol{D}^s \in \mathbb{R}^{p \times q}$ and $p \times q < m \times n$. According to [13], [20] and [21], the projection matrices, represented therefore by $\boldsymbol{\phi}$, need to satisfy the following criterion:

$$\arg\max_{\boldsymbol{\phi}} J(\boldsymbol{\phi}) = \frac{\boldsymbol{\phi}^T \boldsymbol{G_b} \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{G_w} \boldsymbol{\phi}}, \tag{11}$$

which indicates that the optimum projection matrices are the ones that maximize the between class separability and minimize the within class separability.

The optimization process consists, starting from a column projection matrix $\boldsymbol{\phi} = \boldsymbol{U}$, in finding the between and the within class scatter matrices, i.e., $\boldsymbol{G_b} = \boldsymbol{S_b}$ and $\boldsymbol{G_w} = \boldsymbol{S_w}$ through:

$$\boldsymbol{S_b} = \frac{1}{t} \sum_{i=1}^{c} k_i (\bar{\boldsymbol{A}}^i - \bar{\boldsymbol{A}})^T (\bar{\boldsymbol{A}}^i - \bar{\boldsymbol{A}}), \tag{12}$$

and,

$$\boldsymbol{S_w} = \frac{1}{t} \sum_{i=1}^{c} \sum_{j=1}^{k_i} (\boldsymbol{A}_j^i - \bar{\boldsymbol{A}}^i)^T (\boldsymbol{A}_j^i - \bar{\boldsymbol{A}}^i), \tag{13}$$

in which k_i denotes the maximum quantity of elements of class i, and $\boldsymbol{A}_j^i$ indicates the j^{th} element of $\boldsymbol{A}$ that belongs to class i.

According to [13], since the projection matrix $\boldsymbol{U}$ must be constituted of uncorrelated feature vectors, it must be generated from the eigenvalue decomposition:

$$\boldsymbol{S_w^{-1}} \boldsymbol{S_b} = \lambda_j u_j, \tag{14}$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_q$ are the set of eigenvalues associated with the set of column eigenvectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_q\}$ that generates the projection matrix $\boldsymbol{U} \in \mathbb{R}^{n \times q}$.

After finding $\boldsymbol{B} = \boldsymbol{AU}$, in the next step of the optimization $\boldsymbol{\phi} = \boldsymbol{V}$ is found following basically the same approach used to find $\boldsymbol{U}$. However, in such step, the scatter matrices $\boldsymbol{G_b} = \boldsymbol{H_b}$ and $\boldsymbol{G_w} = \boldsymbol{H_w}$ are found following:

$$\boldsymbol{H_b} = \frac{1}{t} \sum_{i=1}^{c} k_i (\bar{\boldsymbol{B}}^i - \bar{\boldsymbol{B}})(\bar{\boldsymbol{B}}^i - \bar{\boldsymbol{B}})^T, \tag{15}$$

and,

$$\boldsymbol{H_w} = \frac{1}{t} \sum_{i=1}^{c} \sum_{j=1}^{k_i} (\boldsymbol{B}_j^i - \bar{\boldsymbol{B}}^i)(\boldsymbol{B}_j^i - \bar{\boldsymbol{B}}^i)^T. \tag{16}$$

After calculating $\boldsymbol{H_b}$ and $\boldsymbol{H_w}$ the eigenvalue decomposition is performed:

$$\boldsymbol{H_w^{-1}} \boldsymbol{H_b} = \epsilon_j v_j, \tag{17}$$

leading to the projection matrix $\boldsymbol{V} \in \mathbb{R}^{m \times p}$, where $\epsilon_1 \geq \epsilon_2 \geq \epsilon_3 \geq \cdots \geq \epsilon_p$ are the set of eigenvalues associated with the set of column eigenvectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_p\}$.

The optimization finally ends when the value obtained in Eq. 11, given the projection matrices $\boldsymbol{V}$ and $\boldsymbol{U}$, stops growing.

III. Proposed Method

In this work, we propose a novel Deep Boltzmann Machine, called 2D-DBM since it utilizes bilinear projections, for robust and fast vehicle classification based on an MLP (Multilayer Perceptron) neural network.

In the proposed classifier, the weights of the MLP neural network are robustly initialized based on the parameters of a trained DBM combined with 2D-LDA projections. The MLP architecture is formed based on the DBM structure formed. After initialization, the MLP parameters are fine-tuned using the conjugate gradient technique.

In the 2D-DBM, the DBM training (together with 2D-LDA projections) is carried out in two stages called Local and Global DBM Pre-Training, likewise in [22]. Such stages, as well as the MLP initialization and fine-tuning, are described in the following subsections.

A. DBM Local Pre-Training

In the first step of the proposed method, the main goal is to train one DBM over each individual patch extracted from the training images in order to improve the local feature extraction of the graphical model. As illustrated in Fig. 3, the local pre-training starts by splitting each sample image (32×32 downsampled color images) into 8×8 patches that represents the local regions of the image. This is done to every

training sample and, following Krizhevsky [22], we used 16 non-overlapping patches plus 9 overlapping patches, resulting in 25 patches per image.

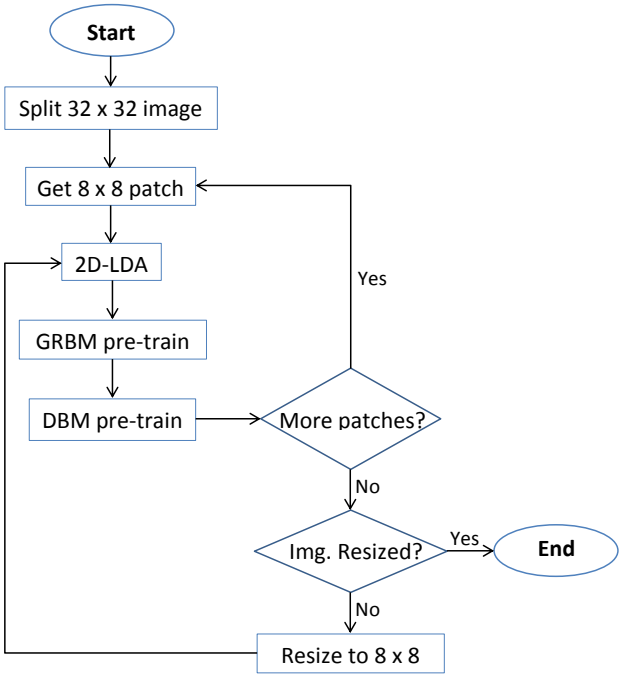


Fig. 3. Local pre-training process: after splitting each original 32×32 training image, each patch is projected in the 2D-LDA space and serves as input to train a Gaussian RBM (GRBM), which is used as interface between the real-valued patch and a binary local DBM. At the end, the whole image is resized to 8×8 pixels and also feeds a GRBM and a local DBM.

Given every group of local patches (extracted from the same 8×8 region of the training images), as shown in Fig. 3, the next step is to apply the bilinear projection technique on them, following [7] and [23], and projection matrices found given all the training patches in the database. Fig. 4 shows our solution to solve the problem of projecting color patches (3 channels - RGB) in the 2D-LDA space and combining the results into a Gaussian Restricted Boltzmann Machine (GRBM) [24] model. For each individual channel, we calculate one pair of projection matrices, $\boldsymbol{U}^{\boldsymbol{8}\ \times\ \boldsymbol{q}^{(f)}}$ and $\boldsymbol{V}^{\boldsymbol{8}\ \times\ \boldsymbol{p}^{(f)}}$, for $f \in \{R, G, B\}$, resulting in $\boldsymbol{W}^{(f)} = \boldsymbol{V}^{(f)\boldsymbol{T}} \otimes \boldsymbol{U}^{(f)}$. Based on this, it is possible to initialize $\mathbf{W}^{\boldsymbol{p}^*\ \times\ \boldsymbol{q}^*}$ since p^* and q^* represent the maximum sizes found during the search for the optimum projection matrices.

In Fig. 4, the blue edges of the GRBM indicate that the weights of such symmetric connections are obtained from the projection matrices generated for each channel of the training patches, obtained from 2D-LDAR (2D-LDA in channel R), 2D-LDAG (bilinear projection in channel G) and 2D-LDAB (2D-LDA in channel B). The black dashed lines indicate the connections of $\mathbf{W}^{\boldsymbol{p}^*\ \times\ \boldsymbol{q}^*}$ that are initialized with value 0.

Using a local GRBM for preprocessing the projected data ensures a more stable convergence for the optimization of the local DBMs. After optimizing each GRBM, the local pre-train

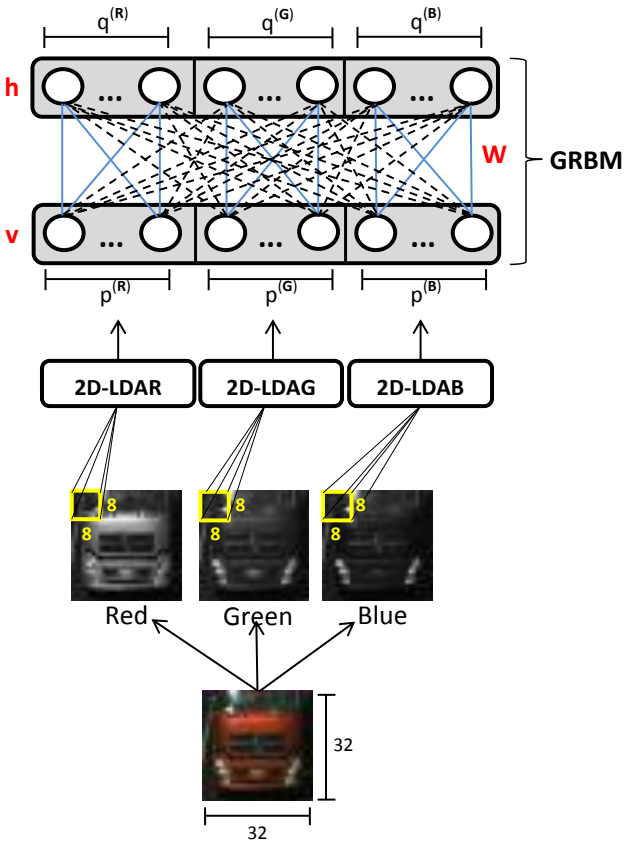


Fig. 4. Color patch projection, GRBM construction and training. Each channel of a training patch is considered separately. For each channel, R, G or B, the patch is projected in the 2D-LDA subspace (found based on all local training patches in the database from the same channel) and the local GRBM is trained: each channel of a given patch feeds part of the visible layer of such GRBM.

process moves on each local DBM optimization, noting that the DBM is placed in the top of the GRBM, i.e., its hidden layer $\boldsymbol{h}$ is considered as the non-sampled version of the first layer of the DBM ($\boldsymbol{v}$).

The local pre-training continues until there is no more patches to be processed for the given training images. After that, the whole 32×32 training images are resized to 8×8 pixels and the same steps applied to their extracted patches are repeated over them.

B. DBM Global Pre-Training

The second step in the DBM training is called global pre-training and consists in combining all the local trained GRBMs and DBMs, where there is no more need to apply the projection technique because we already have scaled the hidden sizes and obtained a good initialization space of parameters $\boldsymbol{\Theta} = \{\mathbf{W}^{\mathbf{1}}, \ldots, \mathbf{W}^{\mathbf{l+1}}, \mathbf{c}^{\mathbf{1}}, \ldots, \mathbf{c}^{\mathbf{l+1}}, \mathbf{b}^{\mathbf{1}}, \ldots, \mathbf{b}^{\mathbf{l+1}}\}$, where $\mathbf{l}$ denotes the maximum amount of patches used in local pre-training phase.

In order to initialize the global DBM, we follow a very similar approach described in [22]. The initialization of the generative and recognition biases, $\mathbf{c}$ and $\mathbf{b}$, respectively, is

performed throw concatenating all the small representations of them in the local DBMs.

Fig. 5 shows the initialization process of parameter $\mathbf{W}$ of the global DBM, that is very similar to the process of initialize the local GRBM in Fig. 4. The main difference is in the concatenation of the local DBMs generated based on the 8×8 patches. For the local DBM trained on the whole reduced training images, the connection values of visible layer nodes with all the other local hidden layers nodes will be an averaged version of its original values.

In Fig. 5, the blue lines represent connections of pre-trained local DBM isolated, black dashed lines represent connections established between hidden and visible nodes from different DBMs initialized with 0 values, and red lines indicate connections between the local DBM pre-trained over the resized version of the entire image. The notation $\mathbf{v^{s(1)}}$ indicates the set of visible nodes coming from the local DBM pre-trained over patch 1 and $\mathbf{h^{s(1)}}$ indicates the corresponding hidden layer of the same local DBM. The $\mathrm{n}+1$ value indicates the resized version of the 32×32 image to 8×8 size. Likewise in Fig. 4, the bias information was visually omitted but it is implicitly in the model.

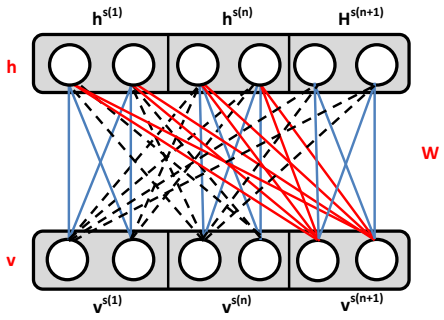


Fig. 5. Global DBM initialization.

C. MLP Initialization and Training

After the DBM pre-training steps, such structure is used, removing its top layer with softmax units (used to improve the DBM feature representation as said), to robustly initialize a MLP neural network, which is then fine-tuned in order to classify unknown images of vehicles into pre-defined classes. The architecture of the formed MLP based on the DBM architecture (with two hidden layers) is shown in Fig. 6. As one can observe, the trained weights ($\boldsymbol{W^1}$, $\boldsymbol{W^2}$ and $\boldsymbol{W^3}$) and biases of the hidden neurons of the DBM ($\boldsymbol{b_1}$ and $\boldsymbol{b_2}$) are used as the initial parameters of the MLP. The symbol ϕ in layers $\mathbf{h^1}$ and $\mathbf{h^2}$ denote the sigmoid function, Eq. 6, applied to the inner product of the output of the previous layer and weights of connections, which is denoted by the symbol Σ.

In Fig. 6, it is also possible to observe the addition of a softmax layer where the abbreviation sft in each node indicates that the input inner product (Σ) is processed by a softmax function, which makes possible to use the cross-entropy function to estimate the error of the MLP predictions and consequently fine-tune such network through backpropagation. As said, in our work the conjugate gradient method was used fo such task. Assuming that the softmax layer produces the normalized distribution of probabilities $S(\cdot)$ for the presented image, which indicates the probability of the vehicle belonging to each pre-defined vehicle class, and that $L(\cdot)$ represents the categorical distribution over the real label of the image (value "1" in the position associated with the correct class of the vehicle and "0" otherwise), the cross-entropy function can be calculated by:

$$Cr(S, L) = -\sum_{i=1}^{|\mathrm{V}|} S(v_i)\ log\ L(v_i), \tag{18}$$

where $|\mathrm{V}|$ represents the cardinality of the set of input images, already preprocessed by the GRBM and submitted to the MLP classifier (v_i indicates each position of S and L).

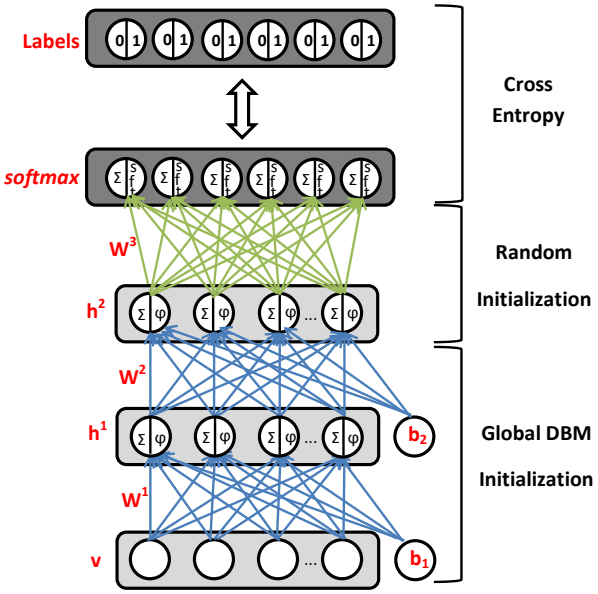


Fig. 6. MLP neural network architecture formed based on the trained DBM. An additional layer with softmax units, one per known class of vehicle, is inserted at the top of the structure to perform classification based on their outcome probabilities given a presented image (after passing through the trained GRBM and the MLP). Such values are compared with the real label of the image through the cross-entropy function to fine-tune the MLP parameters.

IV. Material

The proposed vehicle classification neural network, called 2D-DBM, was assessed on the challenging BIT-Vehicle dataset [6].

For the assessment, the adopted evaluation metric was the average accuracy, traditionally used in literature to compare vehicle classification methods [6].

A. BIT-Vehicle Dataset

The BIT-Vehicle dataset was proposed in [6]. The dataset has a total of 9,905 color (RGB) vehicle images with different sizes: 1600×1200 or 1920×1080 pixels. The images were

captured from two different cameras in different days and from different angles from a real road traffic environment, most of them with frontal view of the vehicles. The luminosity, scale and surface color conditions differ from an image to another. Due to capture delay from the cameras and the large size of some vehicles, in some images it is not possible to observe them as a whole. All these properties make the BIT-Vehicle dataset a challenging database for testing visual vehicle classification methods. Fig. 7 shows some samples of BIT-Vehicle dataset.

Fig. 7. Example of images from the BIT-Vehicle dataset [6].

As illustrated in Fig. 7, the images of vehicles in the BIT-Vehicle dataset [6] are originally distributed into six classes: Bus (555 images), Microbus (878 images), Minivan (474 images), Sedan (5796 images), SUV (1381 images), and Truck (821 images).

In the experiments two subsets of images derived from the BIT-Vehicle dataset were used: Original Images subset and Normalized Images subset.

1) Original Images: This subset is composed by 1200 samples of vehicle images randomly selected from the BIT-Vehicle dataset. Actually, were sampled 1200 images for training and other 1200 for testing, with an equal amount of 200 images per class in both sets. The sampling of the images was performed 5 times, for training and testing sets, and the final accuracies and time results obtained were averaged.

2) Normalized Images: Aiming to overcome the luminosity issues that can be found in most of the images of the BIT-Vehicle dataset, we applied on them two well known histogram equalization techniques based on the HSV (Hue, Saturation and Value) [25] and Lab color spaces [26].

The first equalization technique consists in converting the image from the RGB to the HSV color space and applying an automatic equalization to the histogram of its V component, since it is directly related to the luminosity of the image. This method proved to be efficient to eliminate dark areas that were obstructing the vehicles visualization.

Despite presenting a good performance, the HSV equalization technique, in many cases, was not capable to improve significantly the quality of the images of the BIT-Vehicle dataset. Due to this reason, in such cases, we applied in the vehicle images a second type of normalization based on the Lab space, the Contrast Limited Adaptive Histogram Equalization (CLAHE) technique [27]. To determine wether to use the HSV or conversely the CLAHE normalization, it was defined that the HSV equalization would be applied only when the skewness of the histogram of the V component of the image presented a positive value, greater than 1.0. Fig. 8 shows results of the normalization techniques applied to two vehicle images of the BIT-Vehicle dataset.

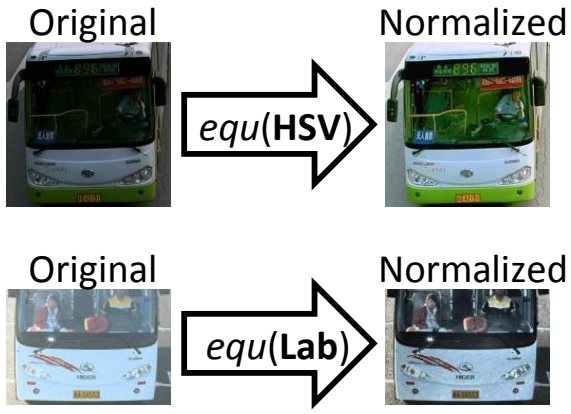


Fig. 8. Images of the BIT database [6] before and after applying one of the two equalizations $equ(\cdot)$. The normalized images present a much better visual aspect.

This subset (Normalized Images) is composed by 1200 samples of normalized vehicle images randomly selected from the BIT-Vehicle dataset. After normalizations of the images, the same sampling approach performed on the Original Images dataset was addopted (also considering 1200 images for training and 1200 for testing).

V. Experimental Results and Discussion

In order to compare the method for vehicle classification based on the new neural network architecture proposed in

this work, 2D-DBM, which utilizes bilinear projections, with the method based on the traditional DBM [12], which does not utilizes bilinear projections, and other method based on CNN [28], several experiments were carried out.

A. *"All Class" Experiments*

The name "All Classes" experiments was given to the group of experiments in which all the six classes of vehicles in the evaluated database were considered.

We performed such experiments configuring the hidden layers for the DBM, that operates without bilinear projections in a complete mode. This means that since the local pre-training of our method used patches with 8×8 pixels, the number of filters of each local hidden layer of DBM was set to 64.

The amount of epochs used for local and global pre-training in the proposed 2D-DBM, as well as in DBM and CNN models, was set to 80, while the learning rates were set to 0.01 and 0.02 for $\mathbf{W}$ and both biases, respectively, during the GRBM pre-training, and 0.1 and 0.2 after that, only for the non 2D-DBM based approach. The same learning rate rule was used during the global pre-training phase, but with the learning rate of $\mathbf{W}$ set to 0.001. Initial momentum was set to 0.5 until fifth epoch and changed to 0.9 after that. For the Mean Field variational inference method, the amount of epochs used was 50, the learning rates were set to 0.0005 and momentums, following the same rule of pre-training stage, were set to 0.1 and 0.5. The quantity of iteration used during the Mean Field positive phase was 30. During the fine-tuning of the MLP, supervised training through conjugate gradient method was performed as said. It was used 50 epochs to train the network over the Original and Normalized Image subsets. For the CNN, the default parameter set up was maintained. In [28] there is a detailed description of such configuration.

In Table I the accuracies (%) obtained on the Original Image (OI) subset and on the Normalized Image (NI) subset are presented.

TABLE I
"ALL CLASSES" EXPERIMENTS - AVERAGE ACCURACIES (%) IN SUBSETS OI AND NI.

METHOD	OI	NI
CNN	70.08 $\pm$ 0.01	69.33 $\pm$ 0.02
DBM	**80.03 $\pm$ 0.02**	**80.62 $\pm$ 0.03**
2D-DBM	77.20 $\pm$ 2.80	78.95 $\pm$ 2.80

In Table II the training times (minutes) of each method are presented.

TABLE II
"ALL CLASSES" EXPERIMENTS - AVERAGE TIME (MINUTES) TO TRAIN THE NETWORK IN OI AND NI SUBSETS.

METHOD	OI	NI
CNN	29m	29m
DBM	44m	93m
2D-DBM	**15m**	**17m**

The results presented in Table I and Table II show that 2D-DBM was superior to CNN in both criteria, accuracy and processing time, in both image subsets, Original Image (OI) and Normalized Image (NI).

These results also show that although being slightly inferior to DBM regarding the criterion accuracy, 2D-DBM was significantly superior to DBM regarding the criterion processing time in both image subsets, Original Image (OI) and Normalized Image (NI).

Regarding the results reported in [6] for BIT-Vehicle dataset (an accuracy rate of 88% using a semisupervised CNN), any comparison is difficult because the authors did not made available the source code of their method. Besides, the training and test subsets of images used in the experiments may not be exactly the same.

B. *"Combined Classes" Experiments*

Since usually only major classes of types of vehicles are considered in real applications of Intelligent Transportation Systems [29], in this group of experiments, that we called "Combined Classes" experiments, we have combined the six classes of the BIT dataset [6] into three main classes: (i) Cars (Sedan & SUV), (ii) Buses (Buses & Micro-buses), and (iii) Trucks (Minivans & Trucks). Minivans original class was grouped into the class Trucks, since both classes represent similar kinds of vehicles in BIT-Vehicle dataset, differing only in dimension. In many cases the minivans in this dataset have truck bodies.

In Table III the accuracies (%) obtained for the Original Image (OI) subset and for the Normalized Image (NI) subset on the "Combined Classes" Experiments are presented. As expected, the accuracies obtained were superior than the ones obtained on the "Combined Classes" experiments.

In Table IV the times spent to train the networks in the "Combined Classes" experiments are presented. It is important to recall that the number of training and test samples per class was increased to 400.

The results obtained in these experiments confirm that 2D-DBM is faster than the other two compared deep learning techniques, DBM and CNN, and also superior to CNN regarding the criterion accuracy.

TABLE III
"COMBINED CLASSES" EXPERIMENTS - AVERAGE ACCURACIES (%) IN SUBSETS OI AND NI.

METHOD	OI	NI
CNN	78.15 $\pm$ 0.02	80.43 $\pm$ 0.01
DBM	**88.45 $\pm$ 0.01**	**89.50 $\pm$ 0.01**
2D-DBM	85.53 $\pm$ 1.16	87.15 $\pm$ 1.43

It was observed during the experiments that the processing time was increased using NI dataset because all the 5 iterations of line search (conjugate gradient method) was executed by the MLP fine tuning step, while a maximum of 3 iterations of line search was executed in the tests with OI dataset.

TABLE IV
"COMBINED CLASSES" EXPERIMENTS - AVERAGE TIME (MINUTES) TO TRAIN THE NETWORK IN OI AND NI SUBSETS.

METHOD	OI	NI
CNN	30m	28m
DBM	45m	1h 35m
2D-DBM	**10m**	**19m**

VI. CONCLUSION

In this work we propose a novel deep learning-based method for visual vehicle classification called 2D-DBM. Besides its superior accuracy levels due to the initialization carried out by means of a DBM, the architecture of the proposed method incorporates bilinear projections in order to obtain efficiency. As well as accuracy, efficiency is an important property of Intelligent Transportation Systems, primarily to meet the online processing requirements.

The 2D-LDA technique plays an important role in the speed up of the pre-training and training phases of the DBM and MLP. In some cases the average time spent to train the bilinear version of the DBMs was 3 times faster than the classical DBM based approaches and 2 times faster than the CNN method in [28]. Besides training speed up, the 2D-LDA technique used in the proposed 2D-DBM method makes the process of defining the architecture more automatic in comparison with CNNs, whose architecture must be reformulated every time the image database changes significantly.

In future work the proposed classifier could be easily extended for other kind of discriminative problems involving images from the real world not related with vehicles and problems involving MLP parameter optimization using techniques like cross validation. It also could be extended for generative problems such as autoencoders [30].

REFERENCES

[1] Y. Baek and W. Kim, "Forward vehicle detection using cluster-based adaboost," *Optical Engineering*, vol. 53, no. 10, pp. 102–103, 2014.

[2] A. Almagambetov, V. S., and M. Casares, "Robust and computationally lightweight autonomous tracking of vehicle taillights and signal detection by embedded smart cameras," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3732–3741, 2015.

[3] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[4] G. Lu, L. Kong, Y. Wang, and D. Tian, "Vehicle trajectory extraction by simple two-dimensional model matching at low camera angles in intersection," *IET Intelligent Transportation Systems*, vol. 8, no. 7, pp. 631–638, 2014.

[5] J. Song, H. Song, and W. Wang, "An accurate vehicle counting approach based on block background modeling and updating," in *2014 7th International Congress on Image and Signal Processing (CISP)*. IEEE, 2014. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6991441

[6] Z. Dong, Y. Wu, M. Pei, and J. Y., "Vehicle type classification using a semisupervised convolution neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 2247–2256, 2015.

[7] H. Wang, Y. Cai, and L. Chen, "A vehicle detection algorithm based on deep belief network," *The Scientific World Journal*, vol. 2014, 2014.

[8] A. Hu, H. Li, F. Zhang, and W. Zhang, "Deep boltzmann machines based vehicle recognition," in *The 26th Chinese Control and Decision Conference (2014 CCDC)*. IEEE, 2014. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6847315

[9] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[10] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K. Müller, Eds. New York, USA: Springer, 2012, pp. 9–48.

[11] I. M. Navon and D. M. Legler, "Conjugate-gradient methods for large-scale minimization in meteorology," *Monthly Weather Review*, vol. 115, 1987.

[12] R. Salakhutdinov and G. E. Hinton, "An efficient learning procedure for deep boltzmann machines," *Neural Computation*, vol. 24, no. 8, pp. 1967–2006, 2012.

[13] J. Yang, D. Zhang, X. Yong, and J.-y. Yang, "Two-dimensional discriminant transform for face recognition," *Pattern recognition*, vol. 38, no. 7, pp. 1125–1129, 2005.

[14] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *SCIENCE*, vol. 220, no. 4598, pp. 671–680, 1983.

[16] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.

[17] M. A. Carreira-Perpiñán and G. E. Hinton, "On contrastive divergence learning," in *aistats05*, Cowell, R. and Ghahramani, Z. Society for Artificial Intelligence and Statistics, 2005. [Online]. Available: http://www.gatsby.ucl.ac.uk/aistats/

[18] M. Welling and G. Hinton, "A new learning algorithm for mean field boltzmann machines," *Artificial Neural NetworksICANN 2002*, pp. 82–82, 2002.

[19] T. Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 1064–1071.

[20] J. Ye, R. Janardan, and Q. Li, "Two-dimensional linear discriminant analysis," in *Proceedings of the 17th International Conference on Neural Information Processing Systems*, ser. NIPS'04. Cambridge, MA, USA: MIT Press, 2004, pp. 1569–1576.

[21] Z. Liang, Y. Li, and P. Shi, "A note on two-dimensional linear discriminant analysis," *Pattern Recogn. Lett.*, vol. 29, no. 16, pp. 2122–2128, dec 2008.

[22] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[23] S. Zhong, Liu Yan, and Liu Yang, "Bilinear deep learning for image classification," in *Proceedings of the 19th ACM International Conference on Multimedia*. ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2072298.2072344

[24] V. Nair and G. E. Hinton, *Implicit Mixtures of Restricted Boltzmann Machines*, Vancouver, B.C., Canada, 2009. [Online]. Available: http://papers.nips.cc/paper/3536-implicit-mixtures-of-restricted-boltzmann-machines.pdf

[25] C. Ware, *Information Visualization: Perception for Design*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.

[26] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[27] K. Zuiderveld, "Graphics gems iv," P. S. Heckbert, Ed. San Diego, CA, USA: Academic Press Professional, Inc., 1994, ch. Contrast Limited Adaptive Histogram Equalization, pp. 474–485. [Online]. Available: http://dl.acm.org/citation.cfm?id=180895.180940

[28] A. Krizhevsky. (2014) cuda-convnet. [Online]. Available: https://code.google.com/p/cuda-convnet/

[29] G. Salvi, "An automated nighttime vehicle counting and detection system for traffic surveillance," in *Proceedings of the 2014 International Conference on Computational Science and Computational Intelligence - Volume 01*, ser. CSCI '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 131–136.

[30] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.