Ceiling Analysis of Pedestrian Recognition Pipeline for an Autonomous Car Application

Henry Roncancio, André Carmona Hernandes and Marcelo Becker
Mobile Robotics Lab (LabRoM)
São Carlos School of Engineering (EESC), University of São Paulo
Av. Trabalhador Sancarlense 400, São Carlos, SP, Brazil.
roncanciovl, andre.hernandes@usp.br, becker@sc.usp.br

Abstract

This paper presents an exploration of the ceiling analysis of machine learning systems. It also provides an approach to the development of pedestrian recognition systems using this analysis. A pedestrian detection pipeline is simulated in order to evaluate this method. The advantage of this method is that it allows determining the most promising pipeline's elements to be modified as a way of more efficiently improving the recognition system. The pedestrian recognition is based on computer vision and is intended for an autonomous car application. A Linear SVM used as classifier enables the recognition, so this development is also addressed as a machine learning problem. This analysis concludes that for this application the more worthy path to be followed is the improvement of the pre-processing method instead of the classifier.

1. Introduction

When it comes to machine learning, ceiling analysis is known as a method to estimate which element of a pipeline machine learning system has a strong influence in the prediction. Likewise, it allows us to estimate which element has a weak influence, and therefore, to limit the effort to improve its performance as it yields no significant change in the final result. This analysis is associated with the concept of ceiling effect, in which an independent variable no longer has an effect on a dependent variable after reaching a certain level, i.e., the ceiling. In the pipeline, this effect can be seen as the impossibility of any increase in the element's performance (the independent variable) to improve the prediction (the dependent variable) because it already reached the "ceiling." In solving the machine learning problem, many resources might be spent trying to improve the elements' performance without actually reaching great results in the entire system. There are several reasons to do a ceiling analysis in any stage of the development: the first is to optimize the resources spent in a task; a further one is to modify the right processing element in order to actually get an upside in the prediction; finally, to determine the benefits of a proposed method before developing it to work automatically. In order to determine the potential upside that might be provided by each processing element, a simulation of each one in its ideal performance is carried out. Thus, it has used a metric, such as the prediction accuracy, to determine the upside in the performance of the entire system when each element is simulated. To do that simulation, the test set is manipulated according to the task of the processing element. In this way, the pipeline is provided with ground-truth labels in each stage. Figure1 presents an example of a machine learning pipeline of three elements and the simulation flow. The main idea behind this procedure is to simulate elements with accuracy of 100%, and therefore, to hopefully increase the performance of the entire system. The simulation is done in a consecutive way, i.e., it begins adding simulated elements from left to right until completing all of them. This simulation is presented by A. Ng at [10].

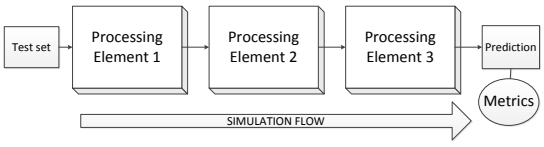

Figure 1. Machine learning pipeline.

This paper presents an approach to the ceiling analysis of the pedestrian recognition problem as a way of improving the detection. As such, the pipeline presented in Figure2 has been proposed in order to determine the potential upsides. These processing elements are the base for many systems, such as those implemented on [11] [4] [2]. However, the pipeline is not limited to these elements, and each one might be split into several sub-elements.

978-1-4673-5648-0/13/$31.00 ©2013 IEEE

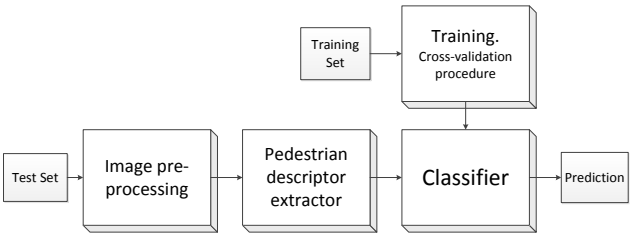

Figure 2. Pedestrian recognition pipeline.

There are four processing elements in this approach: image pre-processing element, pedestrian descriptor extractor, classifier, and training. The first basically extracts a smaller part of the image in which it is more likely to find a pedestrian, because it is easier to recognize the pedestrian when most of the image is only a pedestrian. The next one, the descriptor extractor, is the element that provides the features to the classifier. We have used the histogram of oriented gradients as descriptor; this descriptor yields features more easily recognized when the image is pre-processed because the features are more similar in images that have only pedestrians, i.e., normalized images.

There is also a training element connected to the classifier, as is presented in Figure2, which generates the classifier's model using a supervised learning. This element establishes another line in the pipeline, which will be called the training line. The classifier is the last element in the pipeline and it determines whether the example is a pedestrian or not. Its performance depends on two lines, namely the training line and the processing line. Each line contributes in an independent way to the performance of the system, but they actually use similar methods. The training line implements image normalization and features extraction into the training data the same way as the processing line implements these methods into the test data. Indeed, both lines use the same pedestrian descriptor extractor, i.e., HOG. Even though the image normalization and features extractor methods are in both lines, in this analysis they will only be simulated in the processing line. In the training line will be simulated the training element, which basically simulates the ideal training method considering the ideal training data.

This paper presents a different approach to the analysis of pedestrian recognition systems. In the literature, there are few papers related to the ceiling analysis method for machine learning. This paper is an exploration of this method when it comes to pedestrian recognition. The paper is organized as follows: Section 2 presents the methods used in the processing elements. Section 3 presents the tests. Section 4 describes the ceiling analysis. Finally, Section 5 presents a discussion and Section 6 presents the conclusions.

2. Methods

2.1. Ceiling analysis simulation

The simulation has several stages as described in Figure3. In the first stage is simulated the element that receives the test data. In the following stages, the next elements are added and, it is worthy to note that the elements already simulated continue to be simulated. There is actually no need of simulating the last element; it is assumed that it should provide the remaining upside to achieve an accuracy of 100%.

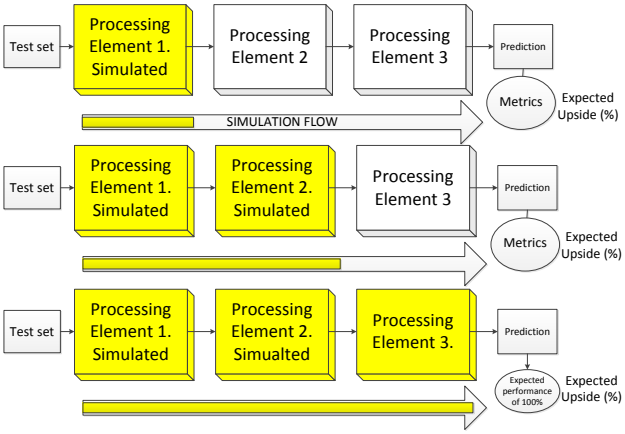

Figure 3. Machine learning pipeline.

The main idea in this analysis is to create a table that specifies the upside in each stage due to the current processing element simulated. An approach to this simulation is presented in Table 1. The first row shows the system accuracy without any simulation. It is the starting point to calculate the upside for the first element. The upside is the result of the subtraction of the current improved accuracy from the previous accuracy. At the end, the simulation will show which element provides the largest upside.

Table 1. Simulation Table for Ceiling Analysis.

Stage	Element	Accuracy	Upside
No simulation	-	Current(%)	-
1st	1st	Improved (%)	(%)
(n-1) stage	(n-1)	Improved (%)	(%)
Last (n)	Last (n)	Expected (100%)	(%)

The ceiling analysis may be seen as a more reliable way for deciding in which element to put a focused effort, because it makes sure that improving that element will actually have a better effect on the performance of the entire system.

2.2. Image pre-processing element

In the processing line, this pre-processing aims to select a smaller part of the image where it is likely to find a pedestrian. There are two well-known approaches to this procedure. The first is looking for pedestrians in the whole image by sliding a search window through it. The second is an approach to regions of interest (ROI), which is based on the data fusion of LIDAR and camera. The laser sensor provides obstacle detection and obstacle's spatial position in the front. This spatial position on the laser's coordinate system is related to the camera's coordinate system in order to find the region of the obstacle in the image. The ROI approach has some advantages with regard to the search window procedure: the processing time is several times shorter than in the search window; it is because in the search case, the window has to be slid through the whole image and resized to look for pedestrians in different scales, which spend more time processing all the windows. Unlike the search window, the data fusion approach tries to find ROI using the laser so as to analyse a smaller part of the image, which reduces the processing time. Another advantage of ROI is that the number of false positives (FPs) is several times smaller than in the search window because it is more likely to detect actual obstacles with the laser sensor. Monteiro *et al*. analysed these elements at [9].

In the training line, there is also a pre-processing procedure to normalize the image; this normalization aims to ensure that the image used for training is mainly a pedestrian. In most of the cases, an expert selects a smaller part of the main image. Unlike the training line, in the processing line the normalization should automatically be carried out in the elements.

2.3. Histogram of oriented gradients (HOG)

This human-shape descriptor created by Dalal [3] is invariant to local geometric and photometric transformations in pedestrian imagery. In this implementation, the resulting descriptor for any image size is a vector $\Re^{81}$ [8]. The descriptor contains the class attributes or features that will be subsequently used to train the classifier.

2.4. Classifier

2.4.1 Support Vector Machine (SVM)

In this paper, an SVM is used as a binary classifier. The SVM implements a mathematical optimization to find a model that best separates the classes; it is done by minimizing the cost function (1) that represents the error between the data and the model.

$$J(\theta)=\tfrac{1}{2}\sum_{j=1}^{n}\theta_j^2+\cdots$$

$$C\sum_{i=1}^{m}[y^{(i)}cost_1(\theta^T x^{(i)})+(1-y^{(i)})cost_0(\theta^T x^{(i)})] \qquad (1)$$

Where:

$J(\theta) : Cost\ function$
$\theta : Parameter\ Vector\ of\ the\ estimated\ hypothesis$
$x^{(i)} : Feature\ vector\ for\ example\ i$
$y^{(i)} : Expected\ output\ for\ example\ i$
$cost_1 : Cost\ for\ positive\ examples$
$cost_0 : Cost\ for\ negative\ examples$
$n : Number\ of\ features$
$m : Number\ of\ examples$
$C : Regularization\ parameter$

2.4.2 Tuned SVM

This classifier is called tuned SVM because it looks for the best Kernel-based SVM parameters that achieve the best F_1 score in the prediction. For the SVM with Linear kernel, the model parameter is C, known as regularization in (1) [6]. The tuning carries out the following steps [7]:

i. *Feature scaling*. First, a feature scaling is accomplished to avoid features in greater numeric ranges dominating those in smaller ranges.

ii. *Looking for the best C parameter in a coarse grid*. C regularizes the model to avoid overfitting the data. In the coarse grid, the parameter varies following rule 2^x for $x = [-6, 5]$, $x \in Z$. A vector of tuning values is then used for the tuning.

iii. *Looking at a fine grid around the coarse tuned value*.

2.5. Training using the Cross-Validation Method (CV)

In order for a model to be statistically well validated its performance should be evaluated on "new data" (different from the training set) several times, otherwise it will show an overoptimistic result. The data are commonly split into two sets, namely training set and cross-validation set. For one split, a simple validation is carried out by calculating on the cross-validation set an estimator of risk *e.g*. accuracy, using the model created with the training set. CV consists in averaging several validations corresponding to different-data splits [1]. Even thought this method is intended to establish the model statistical performance, in this paper a CV is carried out in order to find the best model for our data. Several models are created by varying both the model parameters and the training set and its performance is subsequently evaluated in the cross-validation set.

Because the cross-validation set is used to tune the model, there is a need for another independent data set for correct validation. Therefore, the data is split into three sets,

and the additional set is named test set. A model with high F_1 score in the cross-validation set is considered a good one. The best model is finally validated in the test set in order to calculate a more realistic estimation. This paper uses this CV evaluation to tuning the model just by looking for the model parameters that makes the model predicts better. The type of CV method used is the repeated random sub-sampling validation, i.e., the data are randomly split into training and cross-validation set several times, and for every split both the model accuracy and F_1 score are estimated on the test set. Because the model performance depends on both the model parameters and the training data, a different performance is achieved for each split.

3. Tests and Results

3.1. Data

The database used for this analysis is composed of over 5,000 color images downloaded from INRIA [3]. They were taken under different weather conditions, such as rain and snow/sleet and also with different illumination conditions, including some images taken on night. In all the cases the images were manually normalized.

3.2. Basic Training

In this test the classifier was trained without using the cross-validation procedure. The data sets were taken from the INRIA database. This dataset was originally split into two sets, namely train set and test set (70% and 30% respectively). Table 2 presents a few tests for those original sets. For example, the prediction increased from 82.5% to 91.5% when were used normalized images. In addition, by doing tuning the accuracy reached 93.5%. The SVM was trained using the SVMtrain function of Matlab. The Kernel was a linear function.

3.3. Training using CV procedure

To simulate a set of examples with a larger distribution, this test gathered the whole data from INRIA database and made new sets. The database was split into three sets, 20% of the data was randomly selected for the test set, which was used for validation in the tuning. The other data were consequently used for the CV procedure in order to tune the SVM; Figure4 shows the new data distribution. The positive examples are pedestrian images (70%), and the negative ones are city streets and country images.

This test implemented 10 random sub-sampling validations; the average accuracy reached was 96.9% and the average F_1 score reached was 97.7%. The CV procedure also allowed determining the best model in all validations. This model was subsequently used to make predictions in the test set. The performance of this model is shown in the last row of Table 2.

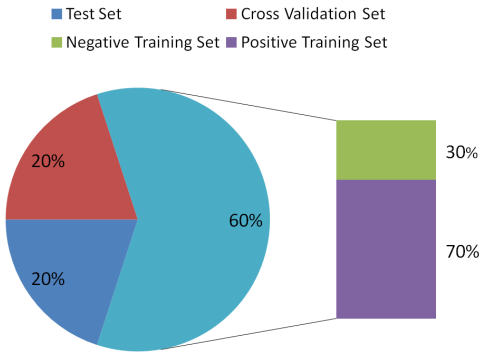

Figure 4. Data Distribution used in CV method.

The increase in the variance of the training examples, i.e. adding new different examples, allowed improving the prediction accuracy from 93.5% to 95.6%, and the F_1 score from 95.3% to 96.7%. With this new data distribution the performance was better. As a way of exploring another classifier' model, an SVM with a radial-based function (RBF) kernel was also tested. The results were very similar; the average accuracy was 95.7% and the average F_1 score was 96.3%.

An advantage of using the CV procedure is that the classifier is trained with a more variable set of examples. In this procedure the sets were randomly arranged, and therefore, they yielded a higher distribution of the observed examples. The tests showed that the larger the variance of this distribution, the better the classifier's model performance.

4. Ceiling Analysis

In the ceiling analysis, several simulations are carried out following the simulation flow. For each simulation, the prediction accuracy is determined and used to calculate the potential upside due to the current element simulated [10]. In this section, it is explained how the recognition pipeline elements are simulated.

Firstly, the pre-processing element is simulated by normalizing the images of the test set. The INRIA database has already normalized the images on that set, so it already yields a way of simulating the element for the whole data set.

Regarding the pedestrian descriptor extractor, its suitable simulation must meet some conditions: to provide the classifier with enough features that are largely similar for examples of the same class and largely different for examples of different classes. It depends on the extractor method itself to meet these conditions; however, the extractor method can be sufficient for a simple database, but for a more complex one it may not. For example, the HOG pedestrian extractor was used in the MIT pedestrian database, in which it provided the classifier with good features so that all the test images were correctly predicted [3]; on the other hand, for a more

Table 2. Metrics calculated for various classifier settings and data.

Classifier Settings and Data description	Accuracy in Test Set	Recall	Precision	F_1 score
Linear SVM in Images without normalization	82.5%	-	-	-
Linear SVM (C = 1 by default) in normalized images	91.5%	90.6%	97.3%	93.8%
Tuned Linear SVM (C = 3.2) in normalized images	93.5%	92.9%	97.9%	95.3%
Tuned Linear SVM using CV procedure in normalized images	95.6%	95.7%	97.7%	96.7%

Table 3. Estimation of potential upside due to each element

Element Simulated	Variable	Method	Accuracy	Potential Upside
		HOG-without normalization	82.5%	
Pre-processing	ROI	Image Normalization	91.5%	9.0%
Descriptor extractor	Descriptor	HOG	*	*
Training	Training Method	Tuned SVM	93.5%	2.0%
Training	Data	New sets and CV procedure	95.6%	2.1%
Classifier	Classifier's model	SVM with linear kernel	100.0%	* 4.4%

complex database, such as the INRIA, the HOG descriptor is seemingly not enough because we could not reach perfect recognition using the same classifier. Consequently, the descriptor should be adjusted according to the scenarios of the test. Other kinds of descriptors could be tested, but this is not the aim of this paper; we are trying to realize what is its function on the entire system in order to simulate it.

The descriptor helps make the task of the classifier easier, because if the features among classes are more largely different they are then more easily separable. According to the machine learning theory, there are some feature manipulations that help the classifier create a better model. For example, by mapping the current features to a higher dimension, in which they are more easily separated (such as the SVM does), it is a way of improving the classifier's performance [10]. Another one is to perform a feature scaling before tuning the classifier's model, such as was mentioned on the tune SVM section. These procedures show that the classifier can modify the features in order to improve its model. Consequently, it is difficult to consider the descriptor extractor as an independent variable to be simulated, because the classifier usually does some additional processing on the features. Some tests intended to evaluate several pedestrian descriptors were able to quantify its performance only after the classifier, by determining the performance in the final prediction [5]. Therefore, there is a near relationship between these processing elements, which makes it difficult to address them separately. This paper does not do a simulation of the descriptor extractor as such; any potential upside provided by the descriptor extractor will be jointly considered when the upside for the classifier is found.

With regard to the training element, there are basically two simulations. The first is the simulation of a larger distribution of training examples, i.e., examples with more differences among them. The idea behind having a larger distribution is to cover a larger quantity of data to be predicted. In a recognition problem, the larger the number of scenarios to be predicted, the larger the number of different training examples. To achieve a good pedestrian recognition system in an autonomous car, the training examples should have as many scenarios of pedestrians in the street as possible. However, to conduct our ceiling analysis, we have limited these scenarios to those covered by the INRIA database. Therefore, to simulate this larger distribution, we have randomly added some examples from the test set to the training set, such as was presented in the CV test section. We have assumed that if there are still some test data that have not been recognized correctly, it is because they have not been included as examples in the training. In this way, to randomly add data to the training set will generate a better training if the data added are new ones. In summary, the larger distribution was simulated by gathering the whole data from INRIA database and creating new sets, subsequently using the CV procedure in these sets to improve the trained model.

The second simulation in the training element is the training method itself. The most important factor is how the hypothesis created after training will generalize new data. Therefore, the right parameter to determine how well the model was trained is the generalization error, i.e., the error in the unseen data. The smallest generalization error is found when the model has the right trade-off between bias and variance, which basically depends on the model itself and its regularization procedure. As shown in the classifier section, there is a regularization parameter in the model that can be tuned to achieve a better regularization. This paper considers the model tuning as a way of achieving the best training method for the SVM classifier we are working with.

The classifier is the last element of the pipeline and, therefore, it does not need simulation itself; it is assumed that if a perfect prediction is required the classifier should provide the last upside to achieve accuracy of 100%. In this

paper, the classifier's model is the method that determines the performance of this element; we have used the SVM with linear kernel as model.

Table 3 shows the processing elements, the independent variables that define the prediction, the method associated with that variable, the prediction result after adding that element in the simulated pipeline, and finally, the potential upside due to that simulation. The first row shows the accuracy of the overall system without any simulation. This test used examples without normalization taken from the INRIA database; the test was done with the original sets. The results show that the largest potential upside is due to the image normalization (9.0%); the training element, taking into account its two variables simulated, shows a potential upside of 4.1%. Both classifier and descriptor extractor together must have a potential upside of 4.4%.

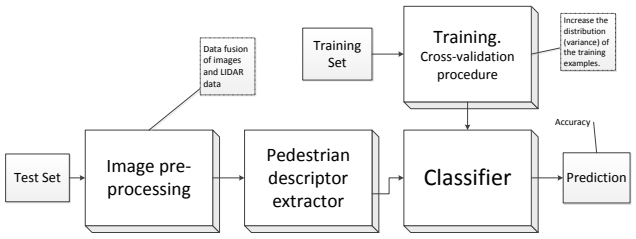

Figure 5. Ceiling analysis results.

5. Discussion

According to our analysis, to select a smaller image where it is likely to find a pedestrian may be a more promising method to improve our system. As mentioned in the preprocessing method, the RIO approach has a few advantages over the sliding window. For future work, a way of getting those ROI is through LIDAR. Because many approaches to autonomous cars already have a LIDAR and a camera, this method is a good option to take advantage of both sensors in the car. LIDAR yields the obstacle's location in the front side, which is correlated with the image. As a result, an obstacle's normalized image is obtained.

In addition, the HOG-Linear SVM method reached an accuracy of 95.6% when the classifier's training was optimized. By tuning it and improving the distribution of the training examples the method obtained an upside of 4.1%.

Figure 5 shows how the processing elements should be modified in order to get a better performance in the prediction. The data fusion and the improvement of the training examples are considered promising approaches.

6. Conclusions

This paper presented an approach to the analysis and development of pedestrian detection systems using the ceiling analysis. It explored the advantage of doing this kind of analysis in the recognition pipeline. It showed that the pre-processing element of our pipeline is the one with the largest potential upside when it comes to improving the prediction, which means it is more worthy to work in this task as it yields a much better improvement in the recognition system. Namely, the image normalization in the processing line. In addition, the improvement of the classifier's training by increasing the quality of the examples and by tuning it becomes another factor to improve the final prediction. The classifier as well as the descriptor is also another important method to be improved.

Acknowledgment

The authors would like to thank São Paulo Research Fundation (FAPESP), Grant # 2011/03986-9, and the Brazilian National Institute of Optics and Photonics (INOF).

References

[1] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

[2] B. Besbes, B. Labbé, A. Rogozan, and A. Bensrhair. SVM-based fast pedestrian recognition using a hierarchical codebook of local features. In *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*, number Mlsp, pages 226–231. IEEE, 2010.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. Ieee, 2005.

[4] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian Detection: An Evaluation of the State of the Art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, July 2012.

[5] D. Gerónimo, A. M. López, A. D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE transactions on pattern analysis and machine intelligence*, 32(7):1239–58, July 2010.

[6] S. Haykin. *Neural networks: a comprehensive foundation*. Pearson Education, Inc., 1999.

[7] C. Hsu, C. Chang, C. Lin, and Others. A practical guide to support vector classification. Technical Report 1, Department of Computer Science National Taiwan University, 2010.

[8] O. Ludwig, D. Delgado, V. Gonçalves, and U. Nunes. Trainable classifier-fusion schemes: an application to pedestrian detection. In *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pages 1–6. IEEE, 2009.

[9] G. Monteiro, P. Peixoto, and U. Nunes. Vision-based pedestrian detection using Haar-like features. *Robotica*, 2006.

[10] A. Ng. Machine Learning, 2012.

[11] L. Oliveira, U. Nunes, and P. Peixoto. On exploration of classifier ensemble synergism in pedestrian detection. *... Systems, IEEE Transactions on*, 11(1):16–27, 2010.