Vision Team Project Proposal : What the hell is that ?

Email Team : vision@chalearn.org
Team : Vincent BOYER *[vincent.boyer1@live.fr]*; Ludovic KUN *[kun.ludovic@yahoo.fr]*;
Qixiang PENG *[kevin.pqx@icloud.com]*; Perceval WAJSBURT *[perceval.wajsburt@gmail.com]*;
Warren PONS *[warren.pons@hotmail.fr]*
Github : *https://github.com/vincentBoyer1/VISION_project*

January 13, 2018

1 Background

Since the end of the 20th century, autonomous vehicles have been debated within the scientific community. Combining artificial intelligence and industrial know-how, they are one of the major objectives of the beginning of the 21th century. We can note for example the GoogleCar project, the Testla Autopilot or the AVA(Autonomous Vehicle for All) project from PSA. However, obstacles remain to be overcome, in particular the behavior of the vehicle in response to its environment of evolution. This step is based on the analysis of its environment and therefore on the entities that make it up. In this challenge, we will therefore study the preliminary stage of the Decision, ie the classification of the detected entities (by the cameras of the vehicle for example). To illustrate this problematic, we propose to study the image source CIFAR-10 which groups entities that can interact with the vehicle environment like animals(cat, horse, dog, ...) and vehicles (bike, car, truck, ...).

2 Material and Methods

2.1 Data description

The dataset used for this project is CIFAR10 [**1**](Figure 1) which is composed of 60 000 images of size 32 x 32 representing 10 labels of animals and means of transport. The datasets is divided as follows: 40 000 for the training step, 10 000 for the testing step and 10 000 for the validation step. Each set is balance distributed (Table 1).

Labels	Train	Test	Validation	Total
airplane	4000	1000	1000	6000
automobile	4000	1000	1000	6000
bird	4000	1000	1000	6000
cat	4000	1000	1000	6000
deer	4000	1000	1000	6000
dog	4000	1000	1000	6000
frog	4000	1000	1000	6000
horse	4000	1000	1000	6000
ship	4000	1000	1000	6000
truck	4000	1000	1000	6000
Total	40000	10000	10000	60000

Table 1: *Distribution of each set : train, validation, set*

2.2 Preprocessing

Before classifying the different images of our dataset, we need to preprocess the images, extracting the features, which will permits us to classify them more precisely. In fact, classifying directly the images will gives us bad results (Table 2), which is why extract features are necessarily. Different features extractors methods are used like HoG (Histogram of Gradient) available on the library skimage in python but the result is pretty bad (Table 3). So we decide to use convolutional neural network used in Deep Learning [**2**]. We use the MobileNet [**3**] architecture of Google on the Keras Framework, and we added some layers so as to reduce the size of the features to 256. We do not want to take features bigger than 256 because the computation time will increase for the different algorithm of

classification. We do not want to reduce too much the features because we want to let the possibility for the student to try to reduce the dimension of the dataset. Regarding the architecture, MobileNet is a good feature extractor because it is a fast and soft model (so our personal computer is powerful enough) with a good accuracy on ImageNet. This is why we use it. We save 256 features for each image in CIFAR10 in the AutoML format.

2.3 Method

We consider our problem as a multi label classification therefore each image is associated to a label value between 0 and 9 included. The final step is to use a machine learning algorithm(using skicit learn for example) so as to be able to predict the class of an image. There are a lot of machine learning algorithm permitting to solve this problem so student have to classify the features with different models and choose the best they obtain. (Naives bayes, SVM, tree, ...). If the student is familiar with the keras framework, he/she can also classify features with a simple neuronal network !
The set of data (features) have 256 dimensions and can be subject to a reduction of dimension in order to limit the training time of the models and to limit the difficulty associated to the Curse of Dimensionality. As we can see on the figure 3, using principal component analysis (PCA), it is possible to lessen the number of features (about 73% is conserved keeping only 10 dimensions, and about 80% is conserved keeping only 20 dimensions). The reduction of dimension is an important pre-processing step and even if it is not essential to get results on this dataset, it permits to save learning time on model which is non-negligible.

To evaluate if the training set is sufficient (enough large), we evaluated the F1-score according to different size of subset. We validate estimations with Stratified K-Fold. Results can be viewed in the figure 2. We can observe an increase until $\tilde{4}$0k entities. After this threshold, the quality of the model stagnate so we can consider that our dataset is enough representative.

Model	Accuracy	Recall	Precision	F1
Tree	0.22	0.22	0.22	0.22
Random Forest	0.21	0.20	0.21	0.15
Naives Bayes Gaussian	**0.27**	**0.28**	**0.27**	**0.25**
Naives Bayes Multinomial	0.24	0.25	0.24	0.22
SVM RBF	0.133	0.289	0.133	0.068

Table 2: *Classification's result without extraction features*

Model	Accuracy	Recall	Precision	F1
Tree	0.254	0.254	0.254	0.254
Random Forest	0.2819	0.314	0.2819	0.231
Naives Bayes Gaussian	**0.4308**	**0.4315**	**0.4308**	**0.421**
Naives Bayes Multinomial	0.4095	0.412	0.4095	0.4048
SVM RBF	0.2698	0.353	0.2698	0.235

Table 3: *Classification's result using HoG as extraction features*

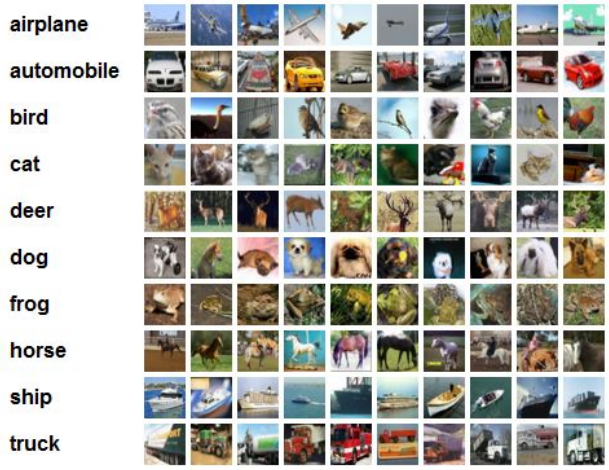

Figure 1: *Representation of the dataset Cifar10*

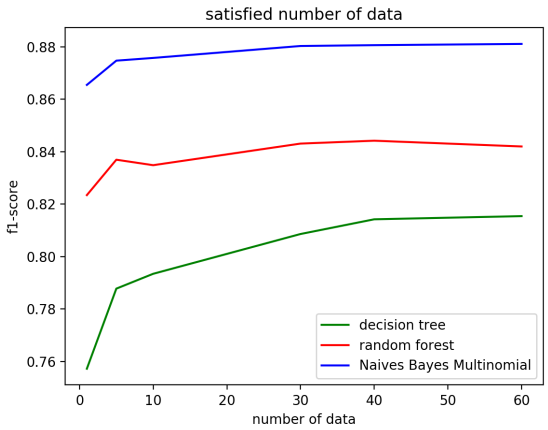

Figure 2: *Representation of the dataset Cifar10*

3 Preliminary results

In this experiment, we tested differents models and compared their performance (Table 4, 5). The different metric used to determine the performance of our models are : accuracy, accuracy per class, recall, precision, F1-score (all with the weighted form to consider balance of label prediction). To validate results for all metrics, we used cross-validation with Stratified K-Fold method to preserve the distribution of the training set

in all subsample. We computed confidence interval (95%) to evaluate the variance of the prediction. We could also used the balanced accuracy metric instead of the regular accuracy but all class of the set are homogenous distributed (same numbers of examples per class) so the result is the same. We note that SVMs models are very efficient on the features extracted by the convolutional neural network and more accuracy equilibrate regarding the accuracy per class (cf Table 5) but it takes some time to construct the model contrary to the others models like tree or random forest which have lowest performances but they are faster to learn. We can also note that tree and random forest have more difficulty predict well a cat contrary to the others models (cf Table 5). However, the computing time of the prediction is pretty fast for all the models. To determine hyperparameters of models, using *Gridsearch* is usefull. For SVM, hyperparameters are [C, kernel, gamma] and, for Random Forest, [criterion, max depth, max number of feature and specificity of leaf and split]. *GridSearch* is time-consuming. We didn't realize this optimization due to strict time period.

The results associated with the learning on the data processed by PCA are available on the figure 3. As we can see, performances are very similar to learning on the initial features (see table 4 and figure 3) when the dimension is superior to 10. The use of PCA (or other method of reducing dimension) is left to the student's sensitivity because this stage of pre-processing is a matter of good understanding about how to learn a dataset. However this step is not mandatory considering the potential lacking experience of the student.

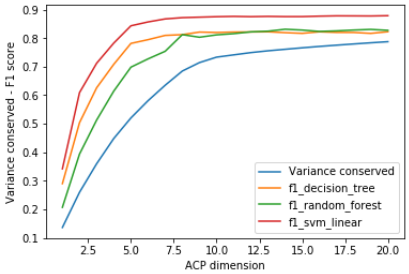

Figure 3: *Graph representing F1 score depending of the number of dimension*

Model	Accuracy	Recall	Precision	F1	Time (sec)
Tree	0.8157 (+/- 0.00437)	0.8157 (+/- 0.00437)	0.8159 (+/- 0.00501)	0.8157 (+/- 0.00471)	16.73
Random Forest	0.8463 (+/- 0.00471)	0.8463 (+/- 0.00471)	0.8448 (+/- 0.00469)	0.8449 (+/- 0.00465)	33.42
Na. Bay. Gaussian	0.8691 (+/- 0.00285)	0.8691 (+/- 0.00285)	0.8719 (+/- 0.00297)	0.8696 (+/- 0.00288)	0.198
Nai. Bay. Multinomial	0.8830 (+/- 0.00242)	0.8830 (+/- 0.00242)	0.8866 (+/- 0.00172)	0.8837 (+/- 0.00213)	**0.06**
SVM rbf	**0.888**	**0.888**	**0.888**	**0.888**	87.49
SVM Linear	0.871	0.872	0.871	0.872	173.15
SVM Poly	0.886	0.887	0.886	0.886	66.24

Table 4: *Classification's result with extraction features on the test set (Note: No cross-validation with SVM model due to computation time)*

Class	Tree	Random Forest	Naives Bayes Gaussian	Naives Bayes Multinomial	SVM rbf	SVM Linear	SVM Poly
airplane	0.821	0.834	0.907	0.924	0.925	0.893	0.918
automobile	0.906	0.93	0.936	0.956	0.952	0.941	0.957
bird	0.732	0.761	0.756	0.825	0.849	0.842	0.85
cat	0.64	0.606	0.772	0.832	0.795	0.779	0.819
deer	0.747	0.774	0.825	0.861	0.871	0.856	0.875
dog	0.73	0.796	0.76	0.768	0.795	0.786	0.773
frog	0.856	0.915	0.882	0.913	0.924	0.914	0.918
horse	0.79	0.851	0.847	0.877	0.899	0.87	0.878
ship	0.871	0.916	0.921	0.942	0.936	0.923	0.937
truck	0.884	0.933	0.925	0.956	0.937	0.915	0.935

Table 5: *Accuracy per class with CNN extraction features on the test set*

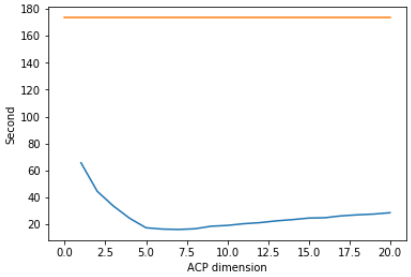

Figure 4: *Graph representing the training time depending of the number of dimension. In blue, the training time of the SVM linear model depending of the number of features, in orange the training time of the SVM linear on the original features(256)*

References

[1] https://www.cs.toronto.edu/ kriz/cifar.html

[2] "CNN Features off-the-shelf: an Astounding Baseline for Recognition", Ali Sharif Razavian et al, 2014.

[3] "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" ,Andrew G. Howard, Menglong Zhu et al, 2017.