Partitioning the Input Domain for Classification

Adrian Rechy Romero*†, Srimal Jayawardena*, Mark Cox* and Paulo Vinicius Koerich Borges*
*Autonomous Systems Laboratory, CSIRO, Australia
†Autonomous Systems Lab, ETH Zurich

***Abstract*— We explore an approach to use simple classification models to solve complex problems by partitioning the input domain into smaller regions that are more amenable to the classifier. For this purpose we investigate two variants of partitioning based on energy, as measured by the variance. We argue that restricting the energy of the input domain limits the complexity of the problem. Therefore, our method directly controls the energy in each partition.**

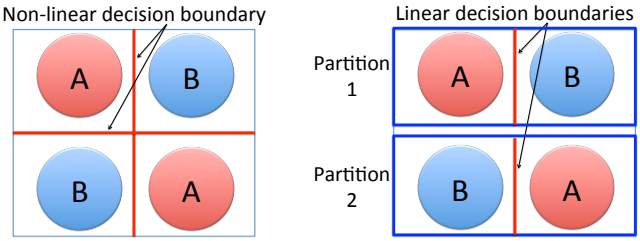
The partitioning methods and several classifiers are evaluated on a road detection application. Our results indicate that partitioning improves the performance of a linear Support Vector Machine and a classifier which considers the average label in each partition, to match the performance of a more sophisticated Neural Network classifier.

I. Introduction

Literature shows that as a classification task becomes more difficult, complex classifiers such as Convolutional Neural Networks [1], [2] and Deep Learning [3], [4] have been used due to their ability to approximate non-linear functions of the input. Rather than using such classifiers, we approach the problem by applying simpler classification models on partitions of the problem's input domain. While a simple model may not be adequate to solve a problem globally, it can be sufficient on local regions of the problem.

Consider the two-class XOR classification problem in Fig. 1a. It cannot be classified correctly using a linear classifier since the decision boundary separating the two classes is non-linear. However, the data can be partitioned as in Fig. 1b such that each partition has a linear decision boundary. Now separate linear classifiers can be applied locally on each partition to correctly classify the data.

This paper explores an energy based partitioning of the input domain with a direct control of the final variance per partition. By defining the maximum variance per partition the partitioning method does not require the number of partitions to be known a priori. We follow the assumption that controlling the energy per partition can be used to modulate the complexity of the problem to make it suitable for different classifiers. For instance, an energy level suitable for a linear Support Vector Machine (SVM) classifier [5] would be one where the data classes in each partition are linearly separable. A simpler classifier that uses the average class label of the partition for its predictions would benefit from an energy level where most of the data in the partition belong to a single class. Experimental results (Sections V and VI) show that input domain partitioning improves the classification results



(a) No partitions. (b) With partitions.

Fig. 1: The figure illustrates a thought experiment on how partitioning helps a linear classifier. Fig. 1a shows a two class classification problem where the decision boundary is non-linear. A single linear classifier would not perform well globally on the entire problem as there is no linear boundary. However, a linear classifier would be adequate locally on partitions of the data (Fig. 1b) where each partition has a linear decision boundary.

of these simple classifiers to a level comparable with a more sophisticated Neural Network classifier.

For a road detection application, partitioning the pixel information was shown to accommodate the variations in pixel colour and position. By partitioning the input domain image-wise we show how an ensemble of pixel-wise classifiers can be trained to account for contextual information in the image. Finally, two stage partitioning (initially image-wise and subsequently pixel-wise) is shown to further increase the performance of the classification.

We make the following contributions:

1) Investigate the effects of unsupervised partitioning prior to classification on different classifiers.
2) Compare two variants of partitioning that directly control the amount of energy per partition and do not require the number of partitions to be known a priori.

II. Related work

Partitioning has been used to solve complex problems using simple models. The input domain has been partitioned by encoding local binary features using a random forest [6] in order to align faces using linear regression. Partitioning has also been used for colour quantisation of images [7] by recursively performing a binary division of the colour palette along the direction of maximum variance until a predetermined number of colours (*i.e.* the number of partitions) is obtained. Our work uses a similar partitioning method but defines the

maximum amount of energy in each partition rather than the number of partitions. Additionally we propose an alternative method for the binary division (Section III-B2).

Standard clustering methods such as k-means and Gaussian Mixture Models (GMMs) [5] could also be used to partition the data. Typical implementations require the number of clusters k to be known beforehand although variants exist [8], [9], [10] that can learn the number of clusters from the data. The clusters generated by these methods depend on the initialisation of the cluster means and the solution is not globally optimal. The partitioning method used in this paper is fully deterministic as it does not require such initialisation and consistently gives the same solution. The number of partitions does not need to be known a priori as the partitioning is based on the expected variance or energy in the resulting partitions.

III. Partitioning

The strategy we employ for partitioning the input domain constructs a binary tree using a recursive procedure. We aim to minimise the energy in each partition, as measured by its variance, in order to reduce the complexity of the input domain for each classifier. Consider the two-dimensional example in Fig. 2. Projections of the data along the direction of maximum variance (Fig. 2a) are compared to a threshold (Fig. 2b) in order to obtain two partitions (Fig. 2c) with less energy. Dividing the input domain along the direction of maximum variance ensures that partitioning is done over the data rather than irrelevant regions of the input domain and that a greater reduction in energy is obtained per division. Fig. 2d shows the tree after the first division. The process is continued recursively until the variance of the leaf nodes reaches an energy level suitable for the classifier. An adequate energy level can be selected through validation techniques such as cross-validation.

Selection of the direction of projection is explained next.

A. Direction of projection

We expect the largest variance reduction by splitting the data along the direction with the highest amount of variance. As used in [11] the direction of projection which maximises the variance of the projected samples is the first eigenvector of the covariance matrix. This can be explained as follows.

Consider a training dataset S where each sample $s \in S$ has a feature vector $\mathbf{x}_s \in \mathbb{R}^{d\times 1}$. We want to divide the set of samples in node n, denoted as $C_n \subseteq S$ into two new nodes $n+1$ and $n+2$, with the sets of samples $C_{n+1} \subset C_n$ and $C_{n+2} \subset C_n$ respectively, such that $C_{n+1} \cup C_{n+2} = C_n$ and $C_{n+1} \cap C_{n+2} = \emptyset$.

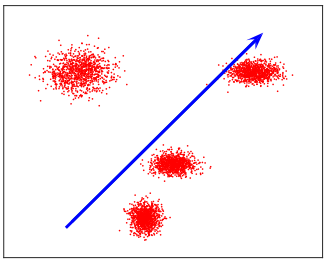
The variance of the node samples along a unit vector $\hat{\mathbf{u}}$ is

$$\sum_{s\in C_n} ((\mathbf{x}_s - \bar{\mathbf{x}}_n)^T \hat{\mathbf{u}})^2 \tag{1}$$

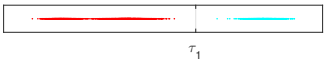
where $\bar{x}_n$ is the mean value of the samples in the node.

Consider the zero-mean data matrix

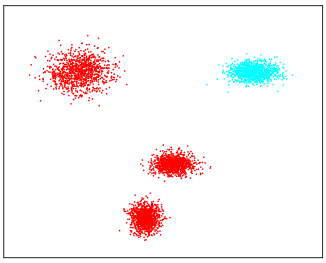
$$\bar{\mathbf{X}}_n = \mathbf{X}_n - \bar{\mathbf{x}}_n \mathbb{1}^T \in \mathbb{R}^{d\times|C_n|} \tag{2}$$



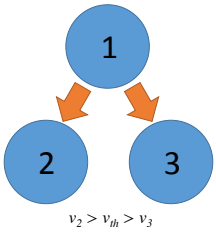
(a) The direction with the maximum variance (Section III-A) is considered in order to partition samples in a node.



(b) Sample projections along the direction in (a) are thresholded using τ_1 (Section III-B) to split the node data.



(c) Partitioned node data corresponding to (b).



(d) Samples in node 1 are partitioned as in (c) to obtain child nodes 2 and 3. The process is continued recursively for leaf node 2 which has a variance $v_2 > v_{th}$.

Fig. 2: A two dimensional example of the partitioning procedure described in Section III.

where the node data matrix

$$\mathbf{X}_n = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|S|}\right] \in \mathbb{R}^{d\times|S|} \tag{3}$$

is normalised row-wise when using features with different ranges. The covariance matrix can be written as $\mathbf{\Sigma} = \bar{\mathbf{X}}_n \bar{\mathbf{X}}_n^T$ and the variance of the projected samples in (1) can expressed in terms of the covariance matrix as $\hat{\mathbf{u}}^T \mathbf{\Sigma} \hat{\mathbf{u}}$. Since $\hat{\mathbf{u}}^T \hat{\mathbf{u}} = 1$ for a unit vector, maximising the variance of the projections is equivalent to maximising the objective

$$\max_{\hat{\mathbf{u}}} \frac{\hat{\mathbf{u}}^T \mathbf{\Sigma} \hat{\mathbf{u}}}{\hat{\mathbf{u}}^T \hat{\mathbf{u}}} \quad \text{s.t.} \quad \hat{\mathbf{u}}^T \hat{\mathbf{u}} = 1 \tag{4}$$

Since $\mathbf{\Sigma}$ is symmetric (4) corresponds to maximising the Rayleigh quotient of Σ and $\hat{\mathbf{u}}$. The solution of (4) is $\hat{\mathbf{u}} = \mathbf{u_1}$, where $\mathbf{u_1}$ is the first eigenvector of $\mathbf{\Sigma}$ corresponding to the largest eigenvalue λ_1 [7]. By considering projections of the node data along $\mathbf{u_1}$

$$\mathbf{p}_n = \bar{\mathbf{X}}_n^T \mathbf{u_1} \in \mathbb{R}^{|C_n| \times 1} \tag{5}$$

partitioning is reduced to a one-dimensional operation.

A projection threshold τ_n is used to split the node data such that projected samples that are greater than the threshold are assigned to the first child and the rest are assigned to the second child, creating the new nodes

$$\begin{aligned} C_{n+1} &= \{s \in C_n : p_{n,s} > \tau_n\} \\ C_{n+2} &= \{s \in C_n : p_{n,s} \leq \tau_n\} \end{aligned} \tag{6}$$

Methods for selecting τ_n are described in Section III-B.

The nodes are split recursively until samples in the leaf nodes have a variance lower than a variance threshold v_{th}. In [7] the first eigenvalue λ_1 of the data is used to represent the variance of a node based on its direction with maximum variance. Alternatively, the trace, which corresponds to the sum of eigenvalues $\sum_j \lambda_j$ gives a description of the energy of the node along all directions.

B. Projection threshold

We consider the following methods for selecting the splitting threshold τ_n in (6).

1) Mean: This is the method used by [7]. The projection threshold is set as the projection of the mean value of the data on $\mathbf{u_1}$. Since the data is centred prior to the projection, the projection threshold $\tau_n = 0 \quad \forall n$.

Partitioning based on the sign of the projection along the first principal direction has been shown [12] to be an upper bound to k-means clustering for $k = 2$. Therefore, using the mean to split the data is an approximation of applying k-means with $k = 2$ successively. However, this splitting method does not guarantee a minimum energy split as shown in Fig. 3. Since the division does not minimise the variance of the child nodes, further divisions are required to reach the variance threshold. A greater number of divisions results in a larger tree that takes longer to traverse for a test sample at runtime.

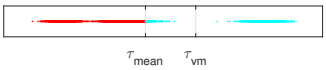
2) Variance minimization: This method checks every possible division of the node data and selects the projection threshold that minimises the combined variance along $\mathbf{u_1}$. The sorted projections $\mathbf{q}_n = \text{sort}(\mathbf{p}_n)$ are searched to find the sample division

$$r = \underset{i}{\text{argmin}} \left(\text{var}(q_{n,(1:i)}) + \text{var}(q_{n,(i+1):end}) \right) \tag{7}$$

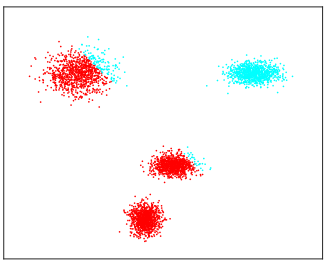
that yields the minimum combined variance. The projection threshold

$$\tau_n = \frac{q_{n,r} + q_{n,(r+1)}}{2} \tag{8}$$

is selected as the mid point between the two samples that encompass the optimal division. This method guarantees that the combined variance of the partitions along the direction of projection is minimised. A better node division tends to achieve a comparable performance with less nodes as seen in the experimental results (Section VI).



(a) The variance based threshold τ_{vm} separates the projections of the data in Fig. 2c better than the mean τ_{mean}.



(b) Sub-optimal partitioning of Fig. 2c when the mean is used to threshold sample projections in (a).

Fig. 3: An example where the variance based threshold (Section III-B2) performs better than the mean based approach (Section III-B1).

C. Characteristics of the partitioning method

The energy based partitioning method has the following characteristics:

1) The size of the tree can be defined based on a variance threshold instead of the number of nodes, as opposed to algorithms such as k-means, which require the number of partitions to be known a priori. This allows partitioning to be done based on the expected energy in a partition.
2) The method does not require an initialisation as with clustering methods such as k-means and GMMs. Therefore the produced results are deterministic.
3) Finding the corresponding partition for a new sample by traversing the node tree has a maximum complexity of $O(\text{depthOfTree} * d)$.

IV. Classification through partitioning

Having partitioned the data, it is possible to perform classification by considering the probability of a class label in each partition. The probability of a test sample s_T having label y given that it belongs to a node n is

$$p(s_T = y|n) = \frac{|C_n \cap Y|}{|C_n|} \tag{9}$$

where Y is the set of training data-samples with label y, $|C_n \cap Y|$ is the number of training samples in the node with label y and $|C_n|$ is the number of training samples in the node. Since the variance within a partition is expected to be low, a given partition is more likely to have data from the same class.

Therefore, considering a maximum likelihood estimate, it is possible to use the average label value of a partition to predict the class of a sample assigned to it at test time. Alternatively other classifiers may be trained on partitions of the input space as shown in Section V-D.

V. Experiments on road detection

Experiments were done on a road detection application to validate the assumption that partitioning reduces the input domain into one that is suitable for the classifier. For this purpose we used the KITTI Vision Benchmark road dataset [13].

Three different approaches for partitioning were tested. Partitioning was first applied directly on the pixel data. Partitioning was also used to divide the dataset image-wise, making groups of similar images. Finally, both partitioning approaches were combined with an image-wise division of the dataset performed before a pixel-wise partitioning was done on each of the image groups.

A. Pixel-wise partitioning

The pixel information of the dataset was partitioned and a different pixel-wise classifier was trained on each partition. The feature vector for partitioning $\mathbf{x}_s = [r, g, b, x, y]^T$ included the three *RGB* channels r, g and b and the pixel coordinates x and y.

The purpose of the *pixel-wise partitioning* is to enhance the classifiers by dividing the classification task into simpler local sub-problems as shown in the example of Fig. 1.

For this experiment, the images were down-sampled into 16×64 pixel images to reduce the time required for training and testing.

B. Image-wise partitioning

As a different approach, the dataset was partitioned into groups of similar images. Each image was used as a sample where the feature vector was built by vectorizing the three *RGB* channels of the entire image such that $\mathbf{x}_s = [r_1, g_1, b_1, ..., r_{xy}, g_{xy}, b_{xy}]^T$. Although partitioning was an image-wise operation, the classifiers still operated on a pixel-wise manner.

Partitioning upon the whole image can be used to aid the pixel-wise classifiers to account for contextual information without increasing their complexity. Since images with a similar colour distribution are grouped into the same partition, a classifier that is trained upon the images from a single partition would specialise on a particular kind of image.

The utility of context information is illustrated in Fig. 4 where a pixel-wise classifier is given the task of determining whether a pixel, given its colour and position, belongs to a circle or a triangle. Given that both the circle and the triangle are green, the pixel information is not enough to determine the shape to which it belongs to. When trained on context-aware partitions, a classifier would only work on images of either triangles or circles and would be able to determine the figure to which the pixel belongs.

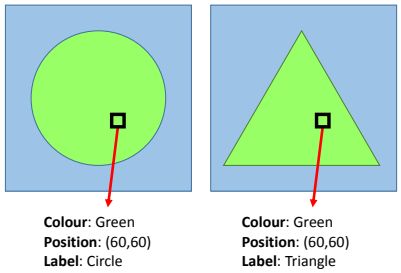


Fig. 4: A classifier based on pixel colour and position is unable to define the object to which a pixel belongs without using context information.

This application for partitioning was evaluated on road detection. The road images were randomly rotated to further increase the complexity of the problem. By changing the location of the road in the image, classifiers that rely on the pixel coordinates as a feature would be encumbered since they would no longer benefit from the fact that the road is usually in the lower part of the image. We expect the partitioning to group images with the same rotation angle together, allowing the pixel coordinates to become a useful feature again.

The images were resized into 10×10 pixel images for the partitioning while the classification was performed on images with a size of 100×100 pixels. Square images were chosen in order to avoid the classifiers from using the different ranges in pixel coordinates to detect the rotation of the image.

C. Two-step partitioning

Pixel-wise partitioning and *image-wise partitioning* were combined in this experiment to integrate the strengths of each individual approach. *Image-wise partitioning* should reduce the variance in rotation and colour distribution of the whole image while the *pixel-wise partitioning* should reduce the variance of the pixel colour and position.

The classifier presented in Section IV was used (further details in Section V-D1) with a variance threshold v_{th} for the *pixel-wise partitioning* set as 1% of the initial variance, while cross-validation was used to find the optimal variance threshold for the *image-wise partitioning*.

D. Classifiers

Three classifiers were tested. The first one based its prediction upon the partitioning alone while the remaining used the three *RGB* channels of each pixel and the pixel coordinates as input parameters to perform a pixel-wise prediction.

1) Mean value of the labels in the partition (Average label method): This classifier finds the partition to which the test sample belongs to and assigns the average value of the labels of the training samples that belong to that partition for the classification. Given binary labels, where road pixels are assigned label 1, this classifier yields the probability of a pixel from being road as stated in Section IV. This classifier

works best when the partitioning reduces the input domain into sections that have a single label per partition.

The classifier changes depending on the data that is being partitioned. In the case of a *pixel-wise partitioning* the classifier would consist of single pixel labels obtained from averaging all the pixel labels from the training data that lie in the same partition. When partitioning on an image-wise manner each partition consists of a collection of images where each image has been assigned a collection of pixel labels that form a labelled image. The classifier for an *image-wise partitioning* averages all labelled images from the training data in the partition and thus, the prediction is also an image.

2) Linear SVM: A linear Support Vector Machine Classifier [5] was also tested. To adapt the data for this classifier the partitioning method should reduce the input domain so that there is a single linear decision boundary on each partition, as shown in the example of Fig. 1. The classifier used is part of the MATLAB Statistics and Machine Learning Toolbox [14]. The command *fitcsvm* was used with the following settings:

- Two-class classification
- Misclassification cost = 1
- No cross validation was performed on the classifier
- Kernel function: linear

3) Neural network: A Neural Network was used to test the effect of partitioning on a higher capacity classifier. This classifier is already non-linear and thus should not benefit from the *pixel-wise partitioning*. Its performance can be set as a benchmark to validate if the previous classifiers were successfully enhanced through partitioning. The Neural Network could still benefit from the infusion of context information through the *image-wise partitioning* since it remains a context-blind classifier due to its pixel-wise behaviour

For this classifier the MATLAB's Neural Network Toolbox was used, with the *patternnet()* and *train()* commands [15]. More specifically, the classifier had the following parameters:

- Hidden layer size = 5
- Dataset was divided randomly. 70% was used for training, 15% for testing and 15% for validation as defined for the default values.
- Training function = *trainscg* (Scaled conjugate gradient backpropagation)

E. Performance Evaluation

Performance evaluation was based on pixel-based metrics. Similar to metrics from [13], the scoring was based on the number of True Positives *TP*, True Negatives *TN*, False Positives *FP* and False Negatives *FN*. Through these parameters it was possible to extract the F_1-measure and the *Accuracy* according to the following equations:

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$F_1 = \frac{(2)Precision\ Recall}{Precision + Recall} \tag{12}$$

TABLE I: Classification results when using a *pixel-wise partitioning*. The first row in each segment corresponds to classification without partitioning. Partitioning improved the classification results for the *Avg. Label* method and the *Linear SVM*. The *Neural Network* classifier did not benefit from partitioning since it already accounts for the non-linearities of the data.

Partitioning	**Classifier**	$\mathbf{v}_{th}$	**No. nodes**	$\mathbf{F}_1$	**Acc.**
None	Avg. label	1	1	0.3064	0.1809
Mean	Avg. label	0.01	13567	**0.8512**	**0.9444**
Var. min.	Avg. label	0.01	11657	0.8467	0.9433
None	Linear SVM	1	1	0.5173	0.7963
Mean	Linear SVM	0.01	13567	**0.8806**	**0.9565**
Var. min.	Linear SVM	0.01	11657	0.8786	0.9557
None	Neural net	1	1	0.8675	0.9518
Mean	Neural net	0.1	133	0.8755	0.9538
Var. min.	Neural net	0.1	123	**0.8763**	**0.9543**

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{13}$$

In the case of the *Average Label* method (Section V-D1) and the *Neural Network* classifier (Section V-D3), where the output is a confidence map, the scoring equations are a function of the classification threshold η. The value of η was chosen to maximise the F_1-measure,

$$\eta = \underset{\gamma}{\operatorname{argmax}}\, F_1(\gamma) \tag{14}$$

Cross-validation was used to optimise upon the amount of partitioning performed for each classifier. The scores shown in Section VI correspond to the average score among the cross-validation buckets that yielded the best F_1-measure.

VI. Results

The following results were obtained for the experiments in Section V.

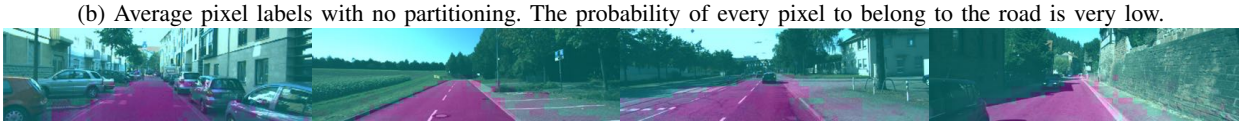
A. Pixel-wise partitioning

The *Average label* and the *Linear SVM* classifiers showed an improvement in performance through a *pixel-wise partitioning* while the performance of the *Neural Network* remained the same. Table I shows the performance of the different classifiers with and without partitioning. The first row in each of the segments corresponds to classification without partitioning. The remaining rows show the classification performance when partitioning with different splitting methods. The best score for each section is highlighted. For a qualitative analysis, Fig. 5 shows several examples where the different classifiers were used with and without partitioning, confirming the quantitative results.

The improvement obtained for the *Average label* method and the *Linear SVM* classifier indicates that the partitioning successfully reduced the input domain to make it suitable for each classifier. The *Average label* method required partitions with a single label while the SVM allowed multiple labels but required a single linear boundary between the classes. The results show that the assumptions were correct since partitioning enhanced the classifiers up to a performance level

(a) Test image.

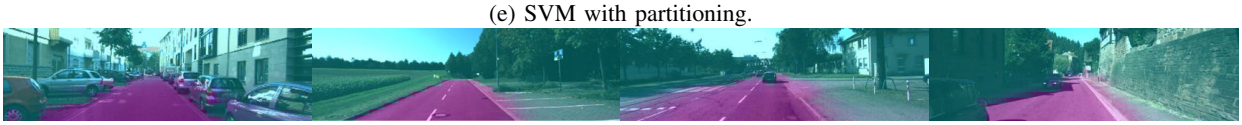
(b) Average pixel labels with no partitioning. The probability of every pixel to belong to the road is very low.



(c) Average pixel labels with partitioning.

(d) SVM with no partitioning.

(e) SVM with partitioning.



(f) Neural Network with no partitioning.

(g) Neural Network with partitioning.

Fig. 5: Pixel-wise partitioning results (Section VI-A).

TABLE II: Classification results when using *image-wise partitioning*. The first row in each segment corresponds to classification without partitioning. The *nan* result for the *SVM* corresponds to a classification where no road labels were predicted. Partitioning improved the performance of the three classifiers.

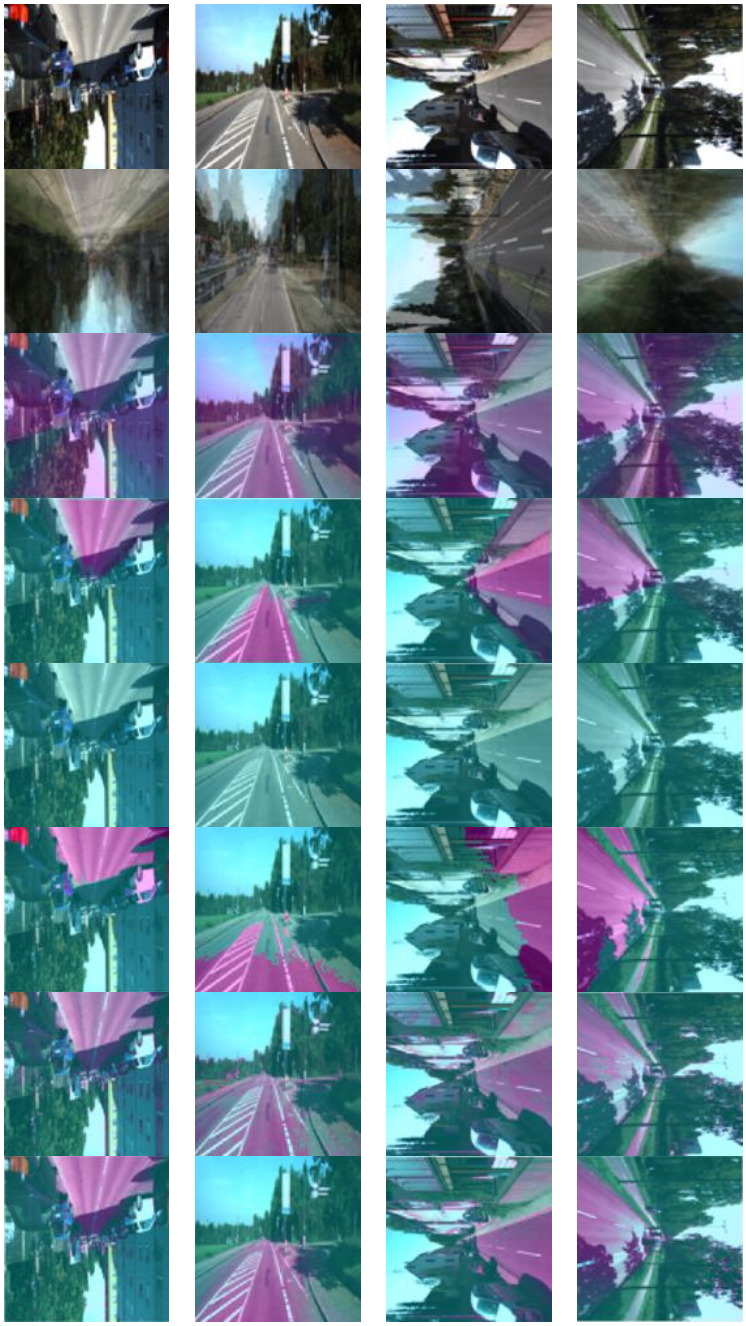
Partitioning	**Classifier**	$\mathbf{v}_{th}$	**No. nodes**	$\mathbf{F}_1$	**Acc.**
None	Avg. label.	1	1	0.3321	0.5136
Mean	Avg. label.	0.2	349	0.6135	**0.8636**
Var. min.	Avg. label	0.3	221	**0.6147**	0.8401
None	Linear SVM	1	1	nan	0.8233
Mean	Linear SVM	0.2	349	**0.5069**	**0.8352**
Var. min.	Linear SVM	0.2	221	0.5056	0.8289
None	Neural net	1	1	0.7197	0.8841
Mean	Neural net	0.7	11	**0.7576**	**0.9102**
Var. min.	Neural net	0.7	11	0.7551	0.9074

that is comparable to the *Neural Network*. Meanwhile, the *Neural Network* did not show significant improvement since it already accounted for the non-linearities of the problem.

B. Image-wise partitioning

Improvement in classification performance was also observed through *image-wise partitioning* (Section V-B). Table II shows the performance of the different classifiers with and without partitioning. Fig. 6 shows a visualization of the effect of partitioning on the different classifiers. For Fig. 6b the partition that corresponded to the test image was found and the training samples that belonged to that partition were averaged, which yielded an image with the same rotation angle and similar colour distribution as the test image, indicating an effective partitioning.

Both the *Average label* method (which in this experiment assigned labelled images rather than pixels) and the *Linear SVM* once again benefited from partitioning. In the case of the *Linear SVM* without partitioning it did not label a single pixel as road, yielding $TP = FP = 0$ which caused (10) and (12) to fail to compute. Meanwhile, the *Neural Network* only had a slight improvement which was not sufficient to achieve the performance from the non-rotated experiment. This was due to the reduction of data samples on each partition since the images of the dataset were only rotated to one angle. Once grouped into similarly rotated images, the amount of images available for each partition's classifier was only a subset of the original dataset.



(a) Test image

(b) Average image from corresponding partition.

(c) Average image labels with no partitioning.

(d) Average image labels with partitioning.

(e) SVM with no partitioning

(f) SVM with partitioning.

(g) Neural Network with no partitioning.

(h) Neural Network with partitioning.

Fig. 6: Image-wise partitioning results (Section VI-B).

C. Partitioning method

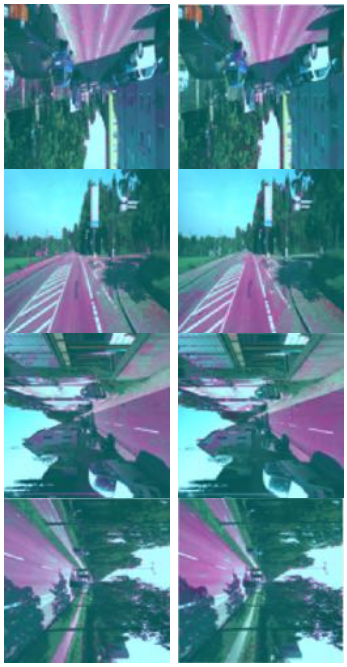
For both the *pixel-wise partitioning* (Table I) and the *image-wise partitioning* (Table II), partitioning was performed with a *Mean*-splitting (Section III-B1) and a *Variance Minimization* approach (Section III-B2) to divide the data of each node. The performance, as given by the F_1-measure and the *accuracy*, did not change much. However, the experiments that used the *Variance Minimization* method required fewer nodes to achieve this score. Having less nodes improves the partitioning efficiency at runtime since a test sample has to traverse a smaller tree to find its corresponding partition.

D. Two-step partitioning

The results of the combination of *image-wise partitioning* with *pixel-wise partitioning* can be seen in Table III. The combination enhanced a simple averaging classifier to a performance level comparable to a *Neural Network* (see Table II). The performance without any partitioning was poor, with the classifier assigning the same label to every pixel. Introducing the *pixel-wise partitioning* increased the performance of the classifier. Furthermore, the combined partitioning increased the performance even further. Although the *image-wise partitioning* divided the dataset into groups of similarly

TABLE III: The table shows the progressive improvement in classification results with *two-step partitioning* for the *Average Label* classifier. The first row corresponds to no partitioning. The second row shows improved results with *pixel-wise partitioning*. The last row shows a further improvement in results with *image-wise partitioning* followed by *pixel-wise partitioning*.

Partitioning	**Image-wise $\mathbf{v}_{th}$**	**Pixel-wise $\mathbf{v}_{th}$**	**$\mathbf{F}_1$**	**Acc.**
None	1	1	0.3003	0.1767
Pixel-wise only	1	0.01	0.6559	0.8625
Both	0.8	0.01	**0.7166**	**0.8969**



(a) *Pixel-wise partitioning* alone.

(b) Image-wise followed by pixel-wise partitioning.

Fig. 7: Two-step partitioning results (Section VI-D).

rotated images, the result still differs from the one shown in Table I due to the reduced number of images trained for each classifier (the groups of images with a same rotation angle have less images than the original dataset in which all the images belonged to the same group).

A visual example of the effect of the addition of the *image-wise partitioning* is shown in Fig. 7. In comparison to using only a *pixel-wise partitioning*, the combination of both partitioning approaches reduced the amount of false negatives, such as misclassified sidewalk pixels which a purely pixel-based classifier could not differentiate from the road.

VII. Conclusion

In this paper we explored the use of input domain partitioning to enhance the performance of simple classification models. We used an energy based partitioning that directly controls the variance per partition, following the assumption that restricting the energy level of a partition modulates the complexity of the problem to make it suitable for the classifier. Experimental results validate the assumptions by improving the performance of a linear Support Vector Machine and a classifier that assigns the average value of the labels in the partition for its predictions. Partitioning enhanced the performance of these classifiers to a level comparable to results obtained using a more sophisticated Neural Network classifier.

The partitioning approach presented in this work does not directly consider the class distribution during partitioning. An interesting future work is to partition in a supervised manner where the division of the data could be based on the distribution of the class labels.

References

[1] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1237.

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 1725–1732.

[3] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[6] S. Ren, X. Cao, Y. Wei, and J. Sun, "Face alignment at 3000 fps via regressing local binary features," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 1685–1692.

[7] M. Orchard and C. Bouman, "Color quantization of images," *Signal Processing, IEEE Transactions on*, vol. 39, no. 12, pp. 2677–2690, Dec 1991.

[8] G. Hamerly and C. Elkan, "Learning the k in k-means," *Advances in neural information processing systems*, vol. 16, p. 281, 2004.

[9] D. Pelleg, A. W. Moore *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters." in *ICML*, 2000, pp. 727–734.

[10] M. Song and H. Wang, "Highly efficient incremental estimation of gaussian mixture models for online data stream clustering," in *Defense and Security*. International Society for Optics and Photonics, 2005, pp. 174–183.

[11] P. Heckbert, *Color image quantization for frame buffer display*. ACM, 1982, vol. 16, no. 3.

[12] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 29.

[13] J. Fritsch, T. Kuhnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, Oct 2013, pp. 1693–1700.

[14] "Matlab statistics and machine learning toolbox," Natick, Massachusetts, 2015a.

[15] "Matlab neural network toolbox," Natick, Massachusetts, 2015a.