Sparse Matrix Classification on Imbalanced Datasets Using Convolutional Neural Networks

Juan C. Pichel[1] and **Beatriz Pateiro-López**[2]

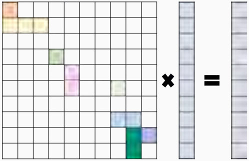
[1]Centro Singular de Investigación en Tecnoloxía Intelixentes (CiTIUS), USC

[2]Departamento de Estatística, Análise Matemática e Optimización, USC

Motivation

Motivation

- Sparse matrix vector multiplication (SpMV) is a key kernel at the core of many scientific and engineering applications



- SpMV performs poorly on multicore architectures since it is a memory-bound operation. Performance depends on both target architecture and sparsity structure
- Many **storage formats** have been proposed: big impact on performance!

Motivation

- Using an inappropriate format could lead to an important degradation in the SpMV performance
- **Automatic selection of the best storage format for sparse matrices on GPUs** is an important and challenging task
- Methodology based on deep learning technologies is introduced
- We address the automatic classification of sparse matrices to select the best SpMV performing storage format on GPUs using Convolutional Neural Networks (CNNs)

Motivation

- The sparsity pattern of the matrices is considered as an image
- RGB color of pixels is used to represent properties of the matrix
- A simple standard CNN architecture as AlexNet is powerful enough to provide very good classification results
- Our methodology can be easily adopted by the research community
- An exhaustive experimental evaluation has been carried out using two different GPUs as target platforms

Background

Background

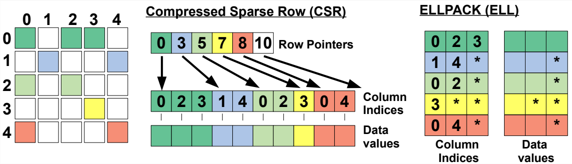
- There exist many different storage formats available. There is not a general storage format that is adequate for all kind of sparse matrices
- These formats differ in the amount of storage required, accessing methods, adaptability to different applications or parallel architectures
- Some formats are only well suited for matrices with a particular pattern: diagonal format (DIA) or block formats (e.g. BELLPACK)

Background

- We focus on formats that are appropriate for matrices coming from different real problems but also efficient for sparse matrix computations
- In particular, we have considered the following formats[1]:
  - Compressed Sparse Row (CSR)
  - ELLPACK (ELL)
  - Hybrid (HYB)
  - Blocked compressed sparse row (BSR)
  - Compressed Sparse Row 5 (CSR5)

[1] Implemented in the NVIDIA cuSPARSE library

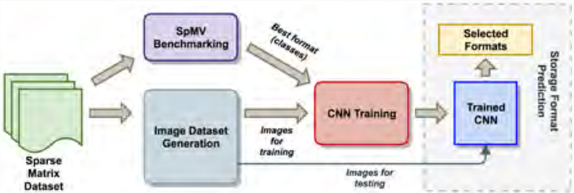
Background



CSR: general-purpose, column indices and nonzeros in two arrays, also array of pointers (offset for each row)
ELL: compress $n \times m$ matrix in a dense $n \times k$ matrix (k = max. number of nonzeros per row). Additional $n \times k$ matrix with column indices
HYB: combines the computation efficiency of ELL with the simplicity and generality of COO (that stores row and column indices explicitly)
BSR: blocked version of the CSR format
CSR5: new storage format based on CSR

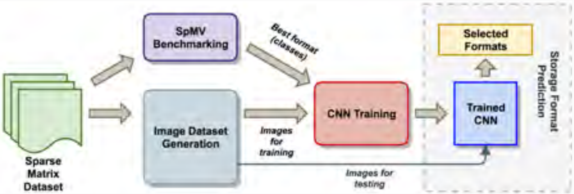
Methodology

Methodology



A large set of sparse matrices coming from different application domains and representing a variety of sparsity patterns is available

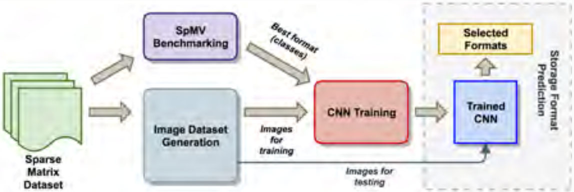
Methodology



A large set of sparse matrices coming from different application domains and representing a variety of sparsity patterns is available

Phase 1: SpMV Benchmarking

- We evaluate the SpMV performance for each matrix considering the different storage formats
- Best format in terms of performance (class/label): ground truth
- We have considered GPUs but methodology is agnostic with respect to the underlying parallel system

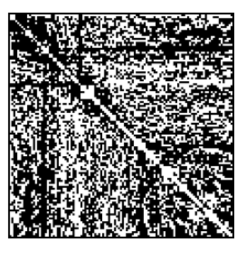


A large set of sparse matrices coming from different application domains and representing a variety of sparsity patterns is available

Phase 2: Image dataset generation

- We consider the sparsity pattern of the matrices as an image
- Naive approach: $n \times m$ matrix is equivalent to a $n \times m$ binary image (problem: input size to a CNN is fixed!)

Methodology



Binary image of 113×113 pixels generated from a 10848×10848 sparse matrix

Methodology

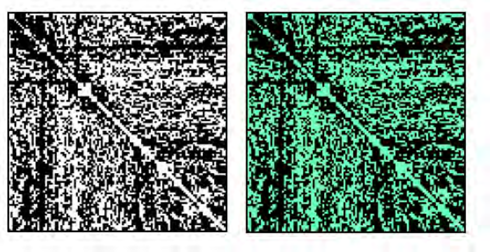
Phase 2: Image dataset generation

- Scaling down the matrix simplifies the appearance of the pattern
- Binary images do not provide competitive results: **additional information**
- We propose to use the RGB channels to code information related to some properties of the original sparse matrix
- The following metrics have been considered:
  0. Matrix size (n): number of rows and columns of the matrix
  1. Average number of nonzeros per row of the matrix (nnz_{row})
  2. Std. deviation of the number of nonzeros per row of the matrix (σ_{row})
  3. Matrix density (ρ): ratio between the number of nonzeros and the number of rows multiplied by the number of columns
  4. Maximum number of nonzeros in a row of the matrix (max_{row})

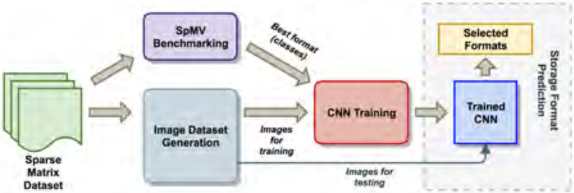
Methodology

Phase 2: Image dataset generation

- Pixels corresponding to empty submatrices are black, that is, RGB is always (0,0,0)
- Pixels representing non-empty submatrices have a different associated RGB color (for each RGB channel is within the interval $[1, 255]$)
- It is possible to use 1, 2 or 3 color channels
- We refer to $\mathbf{R}_x\mathbf{G}_y\mathbf{B}_z$ to indicate that metrics x, y and z correspond to R, G and B values, respectively
- We focus on those most relevant in terms of performance: $\mathbf{R}_1\mathbf{G}_3\mathbf{B}_4$, $\mathbf{R}_0\mathbf{G}_1\mathbf{B}_4$



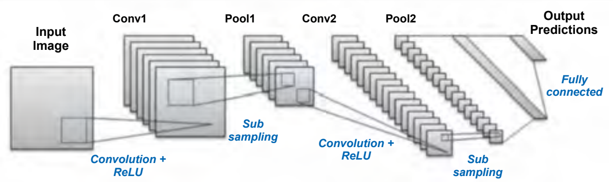
Binary image of 113×113 pixels generated from a 10848×10848 sparse matrix and corresponding $\mathbf{R}_1\mathbf{G}_3\mathbf{B}_4$ image



Phase 3: CNN training

- Input: images labeled with their class (best storage format)
- Dataset divided into training and test sets
- We have used a *k-fold cross-validation* method for model selection and assessment
- Hyperparameter of interest: optimal number of training epochs

Methodology



- Sequence of layers that transform the input image from the original pixel values to the final class scores
- Three types of layers: input layers, feature-extraction layers and classification layers

Methodology

- Many CNN architectures have been proposed
- Most popular are: LeNet, AlexNet, GoogLeNet, VGGNet and ResNet
- We have considered
    - **AlexNet**:
        - 5 convolution layers
        - 3 fully-connected layers
    - **SpNet** (simplified version of AlexNet):
        - 4 convolution layers
        - 2 fully-connected layers

Experimental evaluation

Experimental evaluation

Hardware platforms

We have considered two GPU models, referred as TITANX and QUADRO

Sparse matrix dataset

- We have built a dataset consisting of more than 10k sparse matrices, generated by applying several transformations like cropping to 812 square matrices in the SuiteSparse matrix collection[2]
- Ranges for each matrix feature are wide, which reveal the large diversity of matrices included in the dataset

[2] https://sparse.tamu.edu/

SpMV benchnmarking

- Experiments to measure the performance of the single precision SpMV kernel using 6 storage formats (COO, CSR, HYB, ELL, BSR and CSR5) were conducted on the target GPUs.
- Methods for addressing imbalance are necessary

Class	TITANX	QUADRO
COO	0	0
CSR	1,095 [10.1%]	1,134 [10.5%]
HYB	34 [0.3%]	33 [0.3%]
ELL	525 [4.9%]	531 [4.9%]
BSR	3,552 [32.8%]	3,584 [33.1%]
CSR5	5,616 [51.9%]	5,540 [51.2%]
Total	*10,822*	

Table 1: Distribution of the classes in the matrix dataset.

Experimental evaluation

Networks and training proccess

- We have analyzed and studied two different CNNs to deal with the classification task: AlexNet and SpNet
- To train the networks 80% of the matrices in the dataset are assigned to the training set, while the remainder 20% form the test set
- We have used a k-fold cross-validation method ($k = 5$) to get the optimal number of training epochs
- Other hyperparameters take the default values provided by the DIGITS platform

QUADRO	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.687	***0.872***	0.768	0.683	0.871	0.765	0.678	0.778	0.725	***0.700***	0.869	***0.776***
HYB	-	-	-	0.286	0.667	0.400	0.143	***1***	0.250	***0.429***	***1***	***0.600***
ELL	***0.832***	0.774	0.802	0.738	***0.868***	0.798	0.757	0.786	0.771	0.785	0.840	***0.812***
BSR	0.948	0.948	0.948	***0.964***	0.939	0.951	0.954	0.933	0.943	0.955	***0.950***	***0.953***
CSR5	0.961	0.922	0.941	0.960	0.919	0.939	0.946	***0.927***	0.936	***0.967***	0.924	***0.945***
Global Accuracy	91.9%			91.9%			90.9%			***92.4%***		

TITANX	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.685	0.781	0.730	0.689	0.830	0.753	0.676	0.796	0.731	***0.721***	***0.832***	***0.773***
HYB	-	-	-	0.143	0.200	0.167	0.143	0.333	0.200	***0.286***	***0.400***	***0.333***
ELL	0.829	***0.837***	0.833	0.771	0.818	0.794	***0.838***	0.830	***0.834***	0.790	0.830	0.810
BSR	0.959	0.951	0.955	0.961	0.957	0.959	0.951	0.948	0.949	***0.965***	***0.958***	***0.961***
CSR5	0.953	0.929	0.941	0.959	0.925	0.941	0.953	0.925	0.939	***0.961***	***0.935***	***0.948***
Global Accuracy	91.9%			92.0%			91.6%			***92.8%***		

Table 2: Prediction accuracy of the trained networks considering the two image datasets on QUADRO and TITANX GPUs. Highlighted best results for each format.

Experimental evaluation

QUADRO	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.687	**0.872**	0.768	0.683	0.871	0.765	0.678	0.778	0.725	**0.700**	0.869	**0.776**
HYB	-	-	-	0.286	0.667	0.400	0.143	**1**	0.250	**0.429**	**1**	**0.600**
ELL	**0.832**	0.774	0.802	0.738	**0.868**	0.798	0.757	0.786	0.771	0.785	0.840	**0.812**
BSR	0.948	0.948	0.948	**0.964**	0.939	0.951	0.954	0.933	0.943	0.955	**0.950**	**0.953**
CSR5	0.961	0.922	0.941	0.960	0.919	0.939	0.946	**0.927**	0.936	**0.967**	0.924	**0.945**
Global Accuracy	91.9%			91.9%			90.9%			**92.4%**		

TITANX	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.685	0.781	0.730	0.689	0.830	0.753	0.676	0.796	0.731	**0.721**	**0.832**	**0.773**
HYB	-	-	-	0.143	0.200	0.167	0.143	0.333	0.200	**0.286**	**0.400**	**0.333**
ELL	0.829	**0.837**	0.833	0.771	0.818	0.794	**0.838**	0.830	**0.834**	0.790	0.830	0.810
BSR	0.959	0.951	0.955	0.961	0.957	0.959	0.951	0.948	0.949	**0.965**	**0.958**	**0.961**
CSR5	0.953	0.929	0.941	0.959	0.925	0.941	0.953	0.925	0.939	**0.961**	**0.935**	**0.948**
Global Accuracy	91.9%			92.0%			91.6%			**92.8%**		

Table 2: Prediction accuracy of the trained networks considering the two image datasets on QUADRO and TITANX GPUs. Highlighted best results for each format.

Observations

Good global accuracies for all the cases

QUADRO	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.687	*0.872*	0.768	0.683	0.871	0.765	0.678	0.778	0.725	*0.700*	0.869	*0.776*
HYB	-	-	-	0.286	0.667	0.400	0.143	*1*	0.250	*0.429*	*1*	*0.600*
ELL	*0.832*	0.774	0.802	0.738	*0.868*	0.798	0.757	0.786	0.771	0.785	0.840	*0.812*
BSR	0.948	0.948	0.948	*0.964*	0.939	0.951	0.954	0.933	0.943	0.955	*0.950*	*0.953*
CSR5	0.961	0.922	0.941	0.960	0.919	0.939	0.946	*0.927*	0.936	*0.967*	0.924	*0.945*
Global Accuracy	91.9%			91.9%			90.9%			*92.4%*		

TITANX	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.685	0.781	0.730	0.689	0.830	0.753	0.676	0.796	0.731	*0.721*	*0.832*	*0.773*
HYB	-	-	-	0.143	0.200	0.167	0.143	0.333	0.200	*0.286*	*0.400*	*0.333*
ELL	0.829	*0.837*	0.833	0.771	0.818	0.794	*0.838*	0.830	*0.834*	0.790	0.830	0.810
BSR	0.959	0.951	0.955	0.961	0.957	0.959	0.951	0.948	0.949	*0.965*	*0.958*	*0.961*
CSR5	0.953	0.929	0.941	0.959	0.925	0.941	0.953	0.925	0.939	*0.961*	*0.935*	*0.948*
Global Accuracy	91.9%			92.0%			91.6%			*92.8%*		

Table 2: Prediction accuracy of the trained networks considering the two image datasets on QUADRO and TITANX GPUs. Highlighted best results for each format.

Observations

Best results overall correspond to the $R_0G_1B_4$[3]

[3] Features: size of the matrix, average nº of nonzeros per row and maximum nº of nonzeros in a row of the matrix

Experimental evaluation

QUADRO	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.687	**0.872**	0.768	0.683	0.871	0.765	0.678	0.778	0.725	**0.700**	0.869	**0.776**
HYB	-	-	-	0.286	0.667	0.400	0.143	**1**	0.250	**0.429**	**1**	**0.600**
ELL	**0.832**	0.774	0.802	0.738	**0.868**	0.798	0.757	0.786	0.771	0.785	0.840	**0.812**
BSR	0.948	0.948	0.948	**0.964**	0.939	0.951	0.954	0.933	0.943	0.955	**0.950**	**0.953**
CSR5	0.961	0.922	0.941	0.960	0.919	0.939	0.946	**0.927**	0.936	**0.967**	0.924	**0.945**
Global Accuracy	91.9%			91.9%			90.9%			**92.4%**		

TITANX	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.685	0.781	0.730	0.689	0.830	0.753	0.676	0.796	0.731	**0.721**	**0.832**	**0.773**
HYB	-	-	-	0.143	0.200	0.167	0.143	0.333	0.200	**0.286**	**0.400**	**0.333**
ELL	0.829	**0.837**	0.833	0.771	0.818	0.794	**0.838**	0.830	**0.834**	0.790	0.830	0.810
BSR	0.959	0.951	0.955	0.961	0.957	0.959	0.951	0.948	0.949	**0.965**	**0.958**	**0.961**
CSR5	0.953	0.929	0.941	0.959	0.925	0.941	0.953	0.925	0.939	**0.961**	**0.935**	**0.948**
Global Accuracy	91.9%			92.0%			91.6%			**92.8%**		

Table 2: Prediction accuracy of the trained networks considering the two image datasets on QUADRO and TITANX GPUs. Highlighted best results for each format.

Observations

SpNet clearly outperforms AlexNet when considering $R_0G_1B_4$ images

Experimental evaluation

QUADRO	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.687	***0.872***	0.768	0.683	0.871	0.765	0.678	0.778	0.725	***0.700***	0.869	***0.776***
HYB	-	-	-	0.286	0.667	0.400	0.143	***1***	0.250	***0.429***	***1***	***0.600***
ELL	***0.832***	0.774	0.802	0.738	***0.868***	0.798	0.757	0.786	0.771	0.785	0.840	***0.812***
BSR	0.948	0.948	0.948	***0.964***	0.939	0.951	0.954	0.933	0.943	0.955	***0.950***	***0.953***
CSR5	0.961	0.922	0.941	0.960	0.919	0.939	0.946	***0.927***	0.936	***0.967***	0.924	***0.945***
Global Accuracy	91.9%			91.9%			90.9%			***92.4%***		

TITANX	AlexNet						SpNet					
	$R_1G_3B_4$			$R_0G_1B_4$			$R_1G_3B_4$			$R_0G_1B_4$		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	0.685	0.781	0.730	0.689	0.830	0.753	0.676	0.796	0.731	***0.721***	***0.832***	***0.773***
HYB	-	-	-	0.143	0.200	0.167	0.143	0.333	0.200	***0.286***	***0.400***	***0.333***
ELL	0.829	***0.837***	0.833	0.771	0.818	0.794	***0.838***	0.830	***0.834***	0.790	0.830	0.810
BSR	0.959	0.951	0.955	0.961	0.957	0.959	0.951	0.948	0.949	***0.965***	***0.958***	***0.961***
CSR5	0.953	0.929	0.941	0.959	0.925	0.941	0.953	0.925	0.939	***0.961***	***0.935***	***0.948***
Global Accuracy	91.9%			92.0%			91.6%			***92.8%***		

Table 2: Prediction accuracy of the trained networks considering the two image datasets on QUADRO and TITANX GPUs. Highlighted best results for each format.

Observations

Results for the minority class are poor

Experimental evaluation

Addressing class imbalance

- There are several approaches to deal with the class imbalance problem
- We have considered:
  - oversampling: replicates examples from minority classes to build a more balanced dataset. In our case CSR, ELL and HYB
  - undersampling: randomly remove examples from the majority classes to balance the dataset. Performance results obtained by this technique were not competitive
  - cost-sensitive learning using a weighted loss function

Experimental evaluation

Classes	QUADRO						TITANX					
	Oversampling			Weighted Loss			Oversampling			Weighted Loss		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	***0.744***	0.793	0.773	***0.714***	0.831	0.768	0.712	0.780	0.745	***0.744***	0.791	0.767
HYB	***0.571***	0.800	***0.667***	***0.571***	0.571	0.571	***0.714***	***0.556***	***0.625***	***0.429***	***0.500***	***0.462***
ELL	***0.804***	***0.878***	***0.839***	***0.822***	***0.880***	***0.850***	***0.857***	***0.841***	***0.849***	***0.838***	0.822	***0.830***
BSR	***0.958***	***0.957***	***0.957***	***0.958***	0.948	0.953	0.965	***0.962***	***0.963***	0.965	***0.962***	***0.963***
CSR5	0.957	***0.936***	***0.946***	0.957	***0.931***	0.943	0.949	***0.938***	0.944	0.950	***0.942***	0.946
Global Accuracy	*92.6%*			92.4%			92.5%			92.7%		

Table 3: Prediction accuracy of the trained SpNet networks considering different methods to deal with imbalance using $R_0G_1B_4$ images. Highlighted those values that improve the corresponding results in Table 2

Classes	QUADRO						TITANX					
	Oversampling			Weighted Loss			Oversampling			Weighted Loss		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	**0.744**	0.793	0.773	**0.714**	0.831	0.768	0.712	0.780	0.745	**0.744**	0.791	0.767
HYB	**0.571**	0.800	**0.667**	**0.571**	0.571	0.571	**0.714**	**0.556**	**0.625**	**0.429**	**0.500**	**0.462**
ELL	**0.804**	**0.878**	**0.839**	**0.822**	**0.880**	**0.850**	**0.857**	**0.841**	**0.849**	**0.838**	0.822	**0.830**
BSR	**0.958**	**0.957**	**0.957**	**0.958**	0.948	0.953	0.965	**0.962**	**0.963**	0.965	**0.962**	**0.963**
CSR5	0.957	**0.936**	**0.946**	0.957	**0.931**	0.943	0.949	**0.938**	0.944	0.950	**0.942**	0.946
Global Accuracy	**92.6%**			92.4%			92.5%			92.7%		

Table 3: Prediction accuracy of the trained SpNet networks considering different methods to deal with imbalance using $R_0G_1B_4$ images. Highlighted those values that improve the corresponding results in Table 2

Observations

A better overall behavior is observed

Experimental evaluation

Classes	QUADRO						TITANX					
	Oversampling			Weighted Loss			Oversampling			Weighted Loss		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	***0.744***	0.793	0.773	***0.714***	0.831	0.768	0.712	0.780	0.745	***0.744***	0.791	0.767
HYB	***0.571***	0.800	***0.667***	***0.571***	0.571	0.571	***0.714***	***0.556***	***0.625***	***0.429***	***0.500***	***0.462***
ELL	***0.804***	***0.878***	***0.839***	***0.822***	***0.880***	***0.850***	***0.857***	***0.841***	***0.849***	***0.838***	0.822	***0.830***
BSR	***0.958***	***0.957***	***0.957***	***0.958***	0.948	0.953	0.965	***0.962***	***0.963***	0.965	***0.962***	***0.963***
CSR5	0.957	***0.936***	***0.946***	0.957	***0.931***	0.943	0.949	***0.938***	0.944	0.950	***0.942***	0.946
Global Accuracy	***92.6%***			92.4%			92.5%			92.7%		

Table 3: Prediction accuracy of the trained SpNet networks considering different methods to deal with imbalance using $R_0G_1B_4$ images. Highlighted those values that improve the corresponding results in Table 2

Observations

Oversampling and the cost-sensitive approach obtain similar results

Classes	QUADRO						TITANX					
	Oversampling			Weighted Loss			Oversampling			Weighted Loss		
	Recall	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*	*Recall*	*Prec.*	*F1*
CSR	***0.744***	0.793	0.773	***0.714***	0.831	0.768	0.712	0.780	0.745	***0.744***	0.791	0.767
HYB	***0.571***	0.800	***0.667***	***0.571***	0.571	0.571	***0.714***	***0.556***	***0.625***	***0.429***	***0.500***	***0.462***
ELL	***0.804***	***0.878***	***0.839***	***0.822***	***0.880***	***0.850***	***0.857***	***0.841***	***0.849***	***0.838***	0.822	***0.830***
BSR	***0.958***	***0.957***	***0.957***	***0.958***	0.948	0.953	0.965	***0.962***	***0.963***	0.965	***0.962***	***0.963***
CSR5	0.957	***0.936***	***0.946***	0.957	***0.931***	0.943	0.949	***0.938***	0.944	0.950	***0.942***	0.946
Global Accuracy	***92.6%***			92.4%			92.5%			92.7%		

Table 3: Prediction accuracy of the trained SpNet networks considering different methods to deal with imbalance using $R_0G_1B_4$ images. Highlighted those values that improve the corresponding results in Table 2

Observations

Oversampling does a better job classifying the minority class HYB

Conclusions

Conclusions

- We deal with a problem in the context of the automatic selection of the best SpMV performing storage format for sparse matrices on GPUs
- We consider the sparsity pattern of the matrices as an image, coding several matrix characteristics as the RGB color of the pixels
- A simple trained CNN architecture as AlexNet, without any fine-tuning, achieves very good results. We also introduce a simplified version of AlexNet, called SpNet.
- Several methods to specifically overcome the issues related to training a CNN using imbalance data were studied, obtaining an important overall improvement
- We have compared our approach with a state-of-the-art technique that automatically predicts the best sparse representation using decision trees (without using images). Our proposal outperforms the decision tree model.

Thank you!

beatriz.pateiro@usc.es

J. C. Pichel and B. Pateiro-López.
Sparse matrix classification on imbalanced datasets using convolutional neural networks.
IEEE Access, 7:82377–82389, 2019.