LRP-GUS: A Visual Based Data Reduction Algorithm for Neural Networks

Arnaud Guibert, Christophe Hurter, Nicolas Couellan

LRP-GUS: A visual based data reduction algorithm for Neural Networks

Arnaud Guibert[1][0009−0003−1762−9219], Christophe Hurter[1][0000−0003−4318−6717], and Nicolas Couellan[1,2][0000−0003−3775−1468]

[1] École Nationale de l'Aviation Civile, Université de Toulouse, 7 Avenue Edouard Belin, Toulouse, 31400, France

[2] Institut de Mathématiques de Toulouse, UMR 5219, Université de Toulouse, CNRS, UPS, 118 route de Narbonne, Toulouse, F-31062 Cedex 9, France

Abstract. Deriving general rules to estimate a neural network sample complexity is a difficult problem. Therefore, in practice, datasets are often large to ensure sufficient class samples representation. This comes at the cost of high power consumption and long training time. This paper introduces a novel data reduction method for Deep Learning classifiers, called LRP-GUS, focusing on visual features. The idea behind LRP-GUS is to reduce the size of our training dataset by exploiting visual features and their relevance. The proposed technique is tested on the MNIST and Fashion-MNIST datasets. We evaluate the method using compression rates, accuracy and F_1 scores per class. For instance, our method achieves compression rates of 96.10% for MNIST and 75.94% for Fashion-MNIST, at the cost of a drop of 3% test accuracy for both datasets.

Keywords: Data reduction · Machine Learning · Visual features · XAI

1 Introduction

Training Convolutional Neural Networks (CNNs) requires a lot of resources and time, as feature extraction is integrated in the model itself. One solution to solve this issue is to reduce the training dataset size, using data reduction methods to get a lower number of instances, thus training the model faster without affecting the model's performance and robustness. In a way, this can be seen as an experimental estimation of the CNN sample complexity [2]. In this article, we introduce a new data reduction iterative method called Layerwise Relevance Propagation - Guided Undersampling (LRP-GUS), computing reduced dataset class sizes, based on visual features learnt by the model. Our final goal is to generate a training subset from the original training set: once it is computed, new classifiers with a high enough accuracy can be trained on it, faster than if they were trained on the full training set. We use LRP [3] as our decision basis for class undersampling. This method maps the output of a model to its input space, showing the importance of the input pixels in the model's decision as a saliency map. This saliency map is an image with positive and negative pixel values, showing how each input pixel helped the model to make its decision.

We tested our iterative method on two benchmark image datasets with 10 classes: MNIST [21] (the handwritten digits from 0 to 9) and Fashion-MNIST [32] (various pieces of clothing). Both datasets are composed of 60,000 training images and 10,000 testing images. We trained a distinct CNN architecture for each dataset, to reach an acceptable initial generalization accuracy. With a stop condition of a maximum 3% accuracy drop, our method achieved high training dataset reduction, 96.10% for MNIST and 75.94% for Fashion-MNIST.

This paper is organized as follows: Section 2 lists methods aiming for data reduction and undersampling datasets. Our method and the reason for using visual features are detailled in Section 3. Section 4 describes the experimental framework, and examines the results obtained in this study on the MNIST and Fashion-MNIST datasets. We discuss the results and their limitations in Section 5. Section 6 concludes the paper and discusses potential future work.

2 Related Work

Data reduction is a problem that has been widely documented in the recent years [10, 5]. It consists of selecting instances from the initial training set, based on different metrics. Its end goal is often to find an acceptable trade-off between the accuracy and the compression rate from the full training set. However, some methods focus on staying above a given accuracy at all costs while others focus on compressing the training data as much as possible. While Deep Learning methods can benefit from a good data compression, it can also be detrimental [29, 9] both for the generalization and the convergence speed of the model [1]. Class imbalance methods can also tackle the same issue. They consist in averaging the number of instances per class in a dataset to ensure a balanced class representation, because consequent differences in class representation can hinder the performance of classifiers [15]. Class imbalance can refer to different distributions of datasets, but all of them have an important gap between the minority class and the majority class [10].

Data reduction: Data reduction techniques try to reduce the initial training dataset while keeping the accuracy as high as possible. Researchers have tried various ways to apply this to existing datasets, such as the pure k-Nearest Neighbors selection approach [7], and a few variants [13, 30] focusing on border instances or changing the metric formula. Editing methods focus on cleaning the data, by removing noisy instances and smoothing the inter-class borders [31, 14]. While Euclidean distance is a common choice, some authors have used other distances: MeanShift [6] is an example of such method using dissimilarity measures. Finally, clustering methods are also widely used for data reduction. A few examples are Prototype Selection by Clustering [25] that handles instances clusters based on their homogeneity, Symbolic Nearest Mean Classifier [8] and Reduction through Homogeneous Clusters [26]. All these methods only consider the data and their relative proximity in the data space without looking at the evolution of the model during its training phase. In this article, we fill this gap

by implicitly constructing a metric based on the data and the model. Our goal is to have a better estimate of the sample complexity [2].

Class imbalance: Class imbalance methods can be divided in two groups: oversampling and undersampling methods. We will only focus on undersampling methods since we aim to reduce the dataset size. However, undersampling is not as popular among class imbalance research as oversampling because it tends to degrade the model performance [4]. The most basic method is called random majority undersampling, or RUS. It discards majority classes instances at random until the dataset is more balanced. This method is effective because of the reduced power consumption and training time of the model, but it can perform poorly because of the uncontrolled loss of information from the initial data [12]. One-Sided Selection [19] is an early technique that identifies redundancy of samples close to the class boundaries to remove them. Edited Nearest Neighbor [20] is a technique that detects redundant or insignificant majority samples based on a 3 Nearest Neighbor rule. SMUTE and CSMOUTE [17] are other examples of undersampling techniques, known to work particulary well with complex classifiers like Artificial Neural Networks. Other techniques also use other families of methods like clustering or genetic algorithms to select their instances [33, 22, 18, 11].

XAI methods: eXplainable AI (XAI) methods map the output of a model to its input space. For image classification, they show the importance of the input pixels in the model's decision as a saliency map. Those maps can take multiple forms, depending on the method. Grad-CAM [28] is one such method, outputting a positive saliency map focusing on the important pixels. The intensity of that saliency map shows which pixels contributed the most for the classification. Local Interpretable Model-Agnostic Explanations (LIME) [27] is another method, quantifying the response of the model to a given instance. It trains an explainable model based on random instances chosen around the reference instance. Other methods, like LRP [3] and SHapley Additive exPlanations [23], take a different approach. They output a saliency map with both positive and negative values. The positive values show the inputs helping the classification, while the negative ones show inputs not helping it. To the best of our knowledge, XAI methods are always used as a visual tool for humans to interpret on, and not as a secondary mean to quantitatively optimize a primary objective.

All data reduction methods aim to compress the initial dataset size while keeping an acceptable generalization accuracy. Our method is more focused on the model's reaction to the dataset and tries to balance the dataset using the relevance of the visual features it has learnt. We used LRP to extract the visual feature relevances learnt by the model and assert them. To the best of our knowledge, this is the first algorithm using visual features and XAI methods to reduce the size of a dataset.

3 The LRP-GUS method

LRP-GUS aims to create a training subset that balances the difficulty for a model to learn every class separately. We used a hybrid method based on RUS and XAI methods to build such a dataset balanced with respect to visual features. In this section, we will discuss our XAI method choice, then we will explain our algorithm step by step.

Neural Networks are infamously known for being *black box* algorithms, meaning it is difficult to analyze their decision process when predicting a class for an input. This is because they have a complex decision boundary compared to traditional methods like k-NN or Decision Trees. XAI methods have been developed to understand how a model makes its prediction. They are popular in image classification tasks as they return saliency maps that are human-readable. Details about the goal and output of such methods are explained in Section 2.

Out of all available methods, we focus on Layerwise Relevance Propagation (LRP) because it provides a direct binary criterion that can be easily exploited at a higher level as we propose. Indeed, in the output class LRP maps, the positive pixels are those contributing to the class decision. Grad-CAM does not provide such information, and LIME requires to train one surrogate model for each classifier and is therefore not suitable for the framework we propose. Several LRP decision rules exist focusing on positive or negative activations and/or weights of the network. Each rule specializes in handling input, upper, middle, or lower layers specifically. Such rules include LRP-0, LRP-ϵ, LRP-γ and the z^β-rule. A composite LRP rule [24] was shown to give better explanations on more complex network architectures. However, since our models for Fashion-MNIST and MNIST are relatively small, as described in Section 4.1, we use a pure LRP-ϵ rule (with parameter $\epsilon = 10^{-4}$) for our entire model. An example of LRP outputs on MNIST is shown on Figure 1.

Notations: Let $D = \{(x_i, y_i) \mid 1 \leq i \leq |D|\} \subset \mathbb{R}^d \times \{1, ..., c\}$ be a dataset for supervised classification with c classes. Let $T \subset D$ and $U \subset D$ be a training set and test set respectively. For any pair of sets A and B so that $A \subset B$, we define the compression rate: $r(A, B) = 1 - \frac{|A|}{|B|}$. Let also, for any $I \subset D$, $I^{(k)} = \{(x, y) \in I \mid y = k\}$ be the k-class set of I for a given class $k \in \{1, ..., c\}$. We also define the negative intensity function n for all samples $x \in \mathbb{R}^d$ as the sum of negative values such that $n(x) = \sum_{i=1}^{d} \min(0, x_i)$.

Finally, we define the normalized LRP function. It applies to: an instance from the input space $x \in \mathbb{R}^d$, a class $y \in \{1, ..., c\}$, and a neural network M. The LRP function can therefore be written as: $l_M : (x, y) \mapsto l_M(x, y) \in \mathbb{R}^d$. To compute an average LRP map across various LRP saliency maps, we propose a normalized version of the LRP function above, using absolute maximum normalization. This normalization ensures the sign of the pixels in the LRP maps remains unchanged. Therefore, we define the normalized LRP function as:

$$\hat{l}_M : (x, y) \mapsto \frac{l_M(x, y)}{\max(|l_M(x, y)|)} \tag{1}$$

LRP saliency maps: To illustrate the construction of LRP maps, consider the following example. A model M is trained on the MNIST training dataset, using a network architecture and parameters defined in Section 4.1. We picked two images from the MNIST training set (digit 0 and digit 6) to compute the normalized LRP function $\hat{l_M}$. Figure 1 shows the resulting maps for both digits. The model has learnt a good decision frontier between the class 0 and the other classes as only the LRP heatmap computed using the true class ($y = 0$) displays a lot of positive pixels. The second digit, on the other hand, displays positive visual patterns for almost all classes, not only for its true class 6. This highlights our model can learn clear boundaries for some digits (0 in this case) while the boundaries for others (6 here) are less sharp.

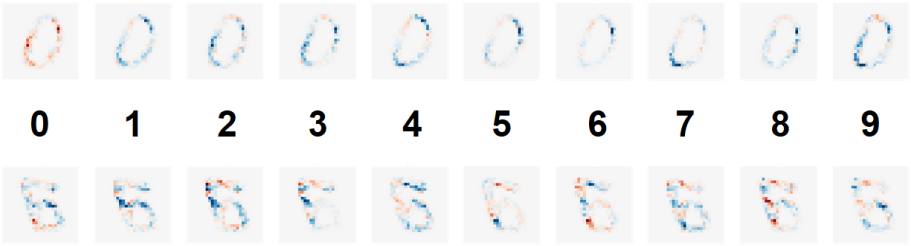

Fig. 1: Normalized LRP maps for 2 MNIST digits. The class y chosen in the LRP formula varies from left (class 0) to right (class 9). In the heatmaps, grey corresponds to scores close to zero, red to positive scores and blue to negative scores. See text for interpretation.

The algorithm: LRP-GUS uses both LRP and Random Undersampling to iterate. The goal is not to pick specific important instances for accuracy, but to find which classes have the most difficult patterns to learn, and thus which ones need more instances to build a balanced CNN.

We start from an initial class proportion $p_k \in \mathbb{N}^c$, where for all $j \in \{1, ..., c\}$, $(p_k)_j$ defines the number of training samples for class j. We use RUS on the training set T to extract an iteration subset I with proportions defined by p_k. We build a neural network classifier M using this extracted dataset I as a training set. We then use the normalized LRP function $\hat{l}_M$ on the training dataset I, with the inputs true class as parameter, and the negative intensity n to compute a vector $f_k \in \mathbb{R}^c$ of scores per class as follows:

$$\forall j \in \{1, ..., c\}, (f_k)_j = n\Big(\frac{1}{|I^{(j)}|} \sum_{(x,y)\in I^{(j)}} \hat{l}_M(x, y)\Big) \tag{2}$$

We then pick a class c_k to increase or decrease its proportion based on the value and sign of the difference between scores at consecutive iterations: $\Delta f_k = f_k - f_{k-1}$. The class c_k must respect a constraint percentage *lim* so that the new

proportion is not too far away from the class proportions mean. We increase or decrease the proportion of that class by a percentage, and end up with a dataset class proportion p_{k+1} for the next iteration, and the scores f_k of the current iteration. If at any point, all the classes violate the constraint, the algorithm stops. Our algorithm outputs a new dataset proportion with less global instances than the initial dataset T, thus reducing the time and energy required to train new models using it. The pseudocode can be found in Table 1.

Table 1: Algorithm pseudocode

```
input: Training set T, Score vector f, Dataset class proportion p,
       Percentages up, down and lim, Max number of iterations Nmax
output: Dataset class proportion p

N ← 0
while N < Nmax do
    if 0 is in p:
        stop condition

    I ← random dataset from T with p
    M ← neural network trained on I
    f_k ← scores of I per class (cf. above)
    c_k ← class whose proportion will be changed

    if (f_k - f)(c_k) > 0 do
        p(c_k) ← (1 - down) * p(c_k)
    else do
        p(c_k) ← (1 + up) * p(c_k)
    f ← f_k
    N ← N + 1

return p
```

4 Results

4.1 Experimental Setup

Experiments were carried out on the MNIST and Fashion-MNIST datasets. MNIST is known for its separability with most handwritten digits sharing common features from their own class, and differences with other classes. Fashion-MNIST is a bit more complex, as some classes visually overlap, making it difficult for a model or even a human to classify them. Each dataset was trained on a separate architecture. Since our goal is to compress the dataset, the chosen architectures and hyperparameters were selected to guarantee both CNNs were "good

enough" classifiers (with a high accuracy on the full training set). While there is room for hyperparameters improvement, this is not the goal of this paper.

Our MNIST network architecture is based on an existing model, LeNet-5 [21]. It is a small and compact CNN. We tuned and changed a few parameters in layers, as described in Table 2. All convolution layers are followed by a ReLU activation function, and the fully connected is followed by a Softmax. Our model uses an Adam optimization [16] algorithm with default α, β_1, β_2 and ϵ parameters. A batch size of 64 was chosen. The model training was stopped after 15 epochs. The Fashion-MNIST architecture introduces more layers, dropout and batch normalization, described in Table 3. The activation functions are the same as in the MNIST model. It uses an Adam optimization algorithm with default parameters, a batch size of 128 and the model training was stopped after 15 epochs.

Table 2: MNIST model

Layer	Size	Kernel	Strides
Input	28*28*1	-	-
Conv2D	26*26*32	3	1
Max Pool	13*13*32	2	2
Conv2D	11*11*64	3	1
Max Pool	5*5*64	2	2
Dropout (.5)	5*5*64	-	-
FC	10	-	-

Table 3: Fashion-MNIST model

Layer	Size	Kernel	Strides
Input	28*28*1	-	-
Conv2D	28*28*16	1	1
Conv2D	28*28*32	3	1
Max Pool	14*14*32	2	2
BatchNorm	14*14*32	-	-
Dropout (.2)	14*14*32	-	-
Conv2D	12*12*64	3	1
Max Pool	6*6*64	2	2
BatchNorm	6*6*64	-	-
Dropout (.2)	6*6*64	-	-
FC	256	-	-
Dropout (.2)	256	-	-
FC	10	-	-

4.2 MNIST

The following initial set of parameters was chosen: score vector $f = 0$ for all classes, dataset class proportion $p = 2000$ for all classes, percentages $up = 5\%$, $down = 20\%$ and $lim = 40\%$ and a maximum number of iterations $Nmax = 1000$. The generalization accuracy with respect to the compression rate is displayed in Figure 2a and the evolution of the dataset proportions p and the F_1 scores per iteration are shown in Figure 2b.

The stop condition for our algorithm is a maximum accuracy drop of 3% compared to the full training set T accuracy. This target is shown as the horizontal black line on Figure 2a. We deduce the *cutoff* iteration i_c from the intersection with the curve. The vertical line $x = i_c$ on Figure 2b gives the *cutoff* dataset T_c and the algorithm final proportion p.

Table 4 shows the repartition of classes in both datasets T and T_c. We can see the introduced imbalance in our subset, while the original training set is

almost balanced. Thus, we trained 20 models on the original training dataset T, our cutoff dataset T_c and on a new dataset T_c (bal), with the same size as T_c, but a forced proportional split of instances between classes. Performance results for those datasets, along with the compression rate are also displayed in Table 4.

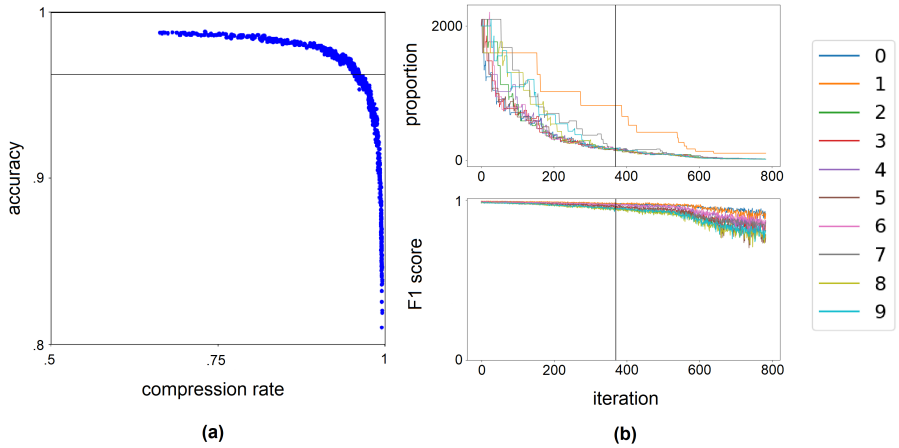

Fig. 2: (a) Evolution of accuracy by compression rate and (b) F_1 scores and class proportions per iteration for our experiments on the MNIST dataset. See text for interpretation.

4.3 Fashion-MNIST

The following initial set of parameters was chosen: score vector $f = 0$ for all classes, dataset class proportion $p = 4000$ for all classes, percentages $up = 5\%$, $down = 20\%$ and $lim = 40\%$ and a maximum number of iterations $Nmax = 500$. The generalization accuracy by the compression rate is displayed in Figure 3a and the evolution of the dataset proportions p and the F_1 scores per iteration are shown in Figure 3b.

The interpretation of the black lines is the same as for MNIST (see Section 4.2). Table 5 shows the repartition of classes in both datasets T and T_c. We also trained 20 models on the original training dataset T, our cutoff dataset T_c and on a new dataset T_c (bal). Performance results on those datasets, along with the compression rate are also displayed in Table 5.

Table 4: Experiment results on MNIST. (a) Proportions of classes in T and T_c. (b) Compression rate, mean and standard deviation for accuracy and F_1 scores. Computed with 20 models for each dataset T, T_c and T_c (bal).

(a)

class	T		T_c	
-	N	%	N	%
0	6923	9.9	185	7.9
1	6742	11.2	819	35.0
2	5958	9.9	173	7.4
3	6131	10.2	173	7.4
4	5842	9.7	162	6.9
5	5421	9.0	152	6.5
6	5918	9.9	162	6.9
7	6265	10.4	154	6.6
8	5851	9.8	180	7.7
9	5949	9.9	180	7.7

(b)

-	T		T_c		T_c (bal)	
-	μ	σ	μ	σ	μ	σ
r	-		96.10		96.10	
accu.	99.24	5e-4	96.35	2e-3	96.63	2e-3
F_1 (0)	99.55	0.10	97.67	0.36	97.75	0.36
F_1 (1)	99.57	0.06	98.50	0.38	98.42	0.23
F_1 (2)	99.17	0.14	95.11	0.49	95.57	0.33
F_1 (3)	99.36	0.10	95.49	0.46	96.35	0.64
F_1 (4)	99.41	0.14	97.52	0.38	97.65	0.38
F_1 (5)	99.02	0.11	95.93	0.62	96.80	0.52
F_1 (6)	99.23	0.13	97.52	0.28	97.65	0.31
F_1 (7)	98.99	0.17	95.32	0.24	95.33	0.63
F_1 (8)	99.09	0.12	94.86	0.41	95.03	0.48
F_1 (9)	98.95	0.19	95.36	0.42	95.62	0.37

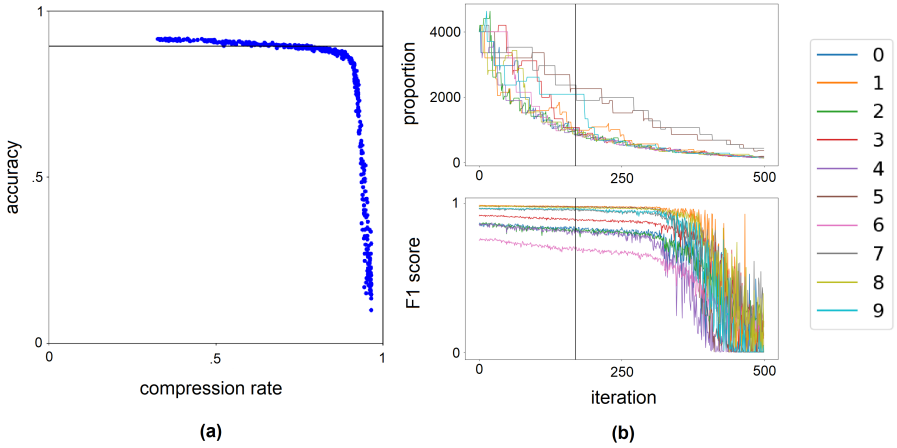

Fig. 3: (a) Evolution of accuracy by compression rate and (b) F_1 scores and class proportions per iteration for our experiments on the Fashion-MNIST dataset. See text for interpretation.

Table 5: Experiment results on Fashion-MNIST. (a) Proportions of classes in T and T_c. (b) Compression rate, mean and standard deviation for accuracy and F_1 scores. Computed with 20 models for each dataset T, T_c and T_c (bal).

(a)

class	T		T_c	
-	N	%	N	%
0	6000	10.0	1068	7.4
1	6000	10.0	1012	7.0
2	6000	10.0	1097	7.6
3	6000	10.0	967	6.7
4	6000	10.0	1130	7.8
5	6000	10.0	3205	22.2
6	6000	10.0	1130	7.8
7	6000	10.0	2685	18.6
8	6000	10.0	1130	7.8
9	6000	10.0	1012	7.0

(b)

-	T		T_c		T_c (bal)	
-	μ	σ	μ	σ	μ	σ
r	-		75.94		75.94	
accu.	92.46	3e-3	88.98	1e-2	89.53	1e-2
F_1 (0)	87.61	0.42	83.22	2.43	83.89	2.66
F_1 (1)	99.02	0.19	98.12	0.18	98.33	0.20
F_1 (2)	87.99	1.25	82.19	2.40	82.72	2.96
F_1 (3)	93.02	0.37	89.36	0.56	90.00	0.62
F_1 (4)	87.94	0.93	81.46	1.75	82.55	2.62
F_1 (5)	98.55	0.24	98.04	0.27	97.50	0.43
F_1 (6)	77.88	0.59	69.55	1.97	71.06	1.83
F_1 (7)	96.82	0.47	95.53	0.86	95.46	0.49
F_1 (8)	98.66	0.15	97.18	0.39	97.50	0.37
F_1 (9)	97.10	0.41	96.03	1.02	96.26	0.29

5 Discussion and Limitations

The proposed method provides a new subset, extracted from the original subset, to train future models on without losing too much accuracy. We can see on Tables 4 and 5 that our algorithm introduces high class imbalance for our experiments on both datasets. This is the reason performance results are confronted to results on a separated balanced final proportion dataset T_c (bal) for both datasets.

For MNIST, the high imbalance shows an increase for class 1, and a decrease for all other classes. However, Table 4 shows that this class imbalance is not a major issue. Indeed, the degradation of the F_1 scores for those classes is not significant compared to other classes like classes 8 and 9. This shows, from our point of view, that class imbalance is not necessarily detrimental in Deep Learning, but may even be a valid idea during training to balance inter-class relevance. For example, the digit 6 proportion was lowered in the dataset from 10% to 7% but its F_1 score barely decreased (-2%). In constrast, other classes like digit 1 need more representation to maintain a good F_1 score. However, the balanced dataset T_c (bal) has better F_1 scores for all classes.

For Fashion-MNIST, the high imbalance shows an increase for classes 5 and 7, and a decrease for all other classes. However, we noticed multiple runs on that dataset returned different proportions. Table 5 shows that the results are more contrasted this time. Classes 5 and 7 were increased in proportion and had their F_1 scores decreased by only 0.5% and 1% respectively, which is successful to us. However, many classes that were reduced in proportion suffered a significant F_1 score drop (-6% for *pullover*, -6.5% for *coat* or -8% for *shirt*). The drop in

proportion of those classes, visually overlapping with each other, was detrimental for their F_1 scores and overall accuracy. Like our MNIST results, the balanced dataset T_c (bal) generalizes a bit better for all classes.

It should also be mentioned from Tables 4 and 5 that there is an increase of the standard deviation σ when reducing the dataset sizes. This should not be a surprise as reducing the number of instances in a class reduces its intraclass variance, thus reducing its generalization to unknown instances.

Both experiments on MNIST and Fashion-MNIST show that our algorithm is aggressive in compression rate, as it tends to force an important degradation of global accuracy and F_1 scores per class. One explanation may be the choice of the *up* and *down* parameters, as picking $+5\%$ and -20% forces a quick degradation of the dataset. Moreover, our stop condition based on accuracy is also limited to datasets and models achieving high initial accuracy. Other possible methods for choosing our *cutoff* iteration are still under investigation.

6 Conclusion

Our method, LRP-GUS, extracts a training subset that reflects more homogeneous visual features learnt across all classes of any given dataset. We used an XAI method, LRP, to assess the visual features and select the class proportions that should be modified. To the best of our knowledge, this is the first data reduction method exploiting such visual features. LRP-GUS achieved high compression rates, 96.10% for MNIST and 75.94% for Fashion-MNIST, at the cost of a drop of 3% test accuracy for both datasets. This shows that deep learning models can be trained on extracted imbalanced subsets while controlling a good trade-off with the predictive performance. This is encouraging as some datasets come with a natural high imbalance (fraud detection, meteo prediction, ...).

As our algorithm is highly dependent of the choice of the *cutoff* iteration, its choice is crucial. In the proposed work, it was selected as corresponding to a drop of 3% accuracy. More clever choices based on automatic detection of regime shift in the accuracy-compression rate graph may be more appropriated. The current non-refined choice of *cutoff* introduces extra class imbalance that may explain the contrasted results when comparing the final reduced dataset and a random balanced dataset with the same compression rate. We believe visual features can give a good explanation about a model's performance, thus allowing to choose instances and train a better model. The results displayed in this paper show the viability of our method in the form of preliminary results. Further investigation will focus on making a proper comparison with data reduction methods. In addition, future work could also investigate other XAI methods beyond LRP like LIME, or by extension SHAP.

References

1. Anand, R., Mehrotra, K.G., Mohan, C.K., Ranka, S.: An improved algorithm for neural network classification of imbalanced training sets. IEEE Transactions on Neural Networks **4**(6), 962–969 (1993)

2. Anthony, M., Bartlett, P.L., Bartlett, P.L., et al.: Neural network learning: Theoretical foundations, vol. 9. cambridge university press Cambridge (1999)
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015)
4. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD explorations newsletter **6**(1), 20–29 (2004)
5. Buda, M., Maki, A., Mazurowski, M.A.: A systematic study of the class imbalance problem in convolutional neural networks. Neural networks **106**, 249–259 (2018)
6. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE transactions on pattern analysis and machine intelligence **17**(8), 790–799 (1995)
7. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE transactions on information theory **13**(1), 21–27 (1967)
8. Datta, P., Kibler, D.: Learning symbolic prototypes. In: ICML. pp. 75–82 (1997)
9. Estabrooks, A., Jo, T., Japkowicz, N.: A multiple resampling method for learning from imbalanced data sets. Computational intelligence **20**(1), 18–36 (2004)
10. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE transactions on pattern analysis and machine intelligence **34**(3), 417–435 (2012)
11. Ha, J., Lee, J.S.: A new under-sampling method using genetic algorithm for imbalanced data classification. In: Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication. pp. 1–6 (2016)
12. Hasanin, T., Khoshgoftaar, T.: The effects of random undersampling with simulated class imbalance for big data. In: 2018 IEEE international conference on information reuse and integration (IRI). pp. 70–79. IEEE (2018)
13. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification and regression. Advances in neural information processing systems **8** (1995)
14. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: A review. IEEE Transactions on pattern analysis and machine intelligence **22**(1), 4–37 (2000)
15. Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. Intelligent data analysis **6**(5), 429–449 (2002)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Koziarski, M.: Csmoute: Combined synthetic oversampling and undersampling technique for imbalanced data classification. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2021)
18. Krawczyk, B., Galar, M., Jeleń, Ł., Herrera, F.: Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. Applied Soft Computing **38**, 714–726 (2016)
19. Kubat, M., Matwin, S., et al.: Addressing the curse of imbalanced training sets: one-sided selection. In: Icml. vol. 97, p. 179. Citeseer (1997)
20. Laurikkala, J.: Improving identification of difficult small classes by balancing class distribution. In: Conference on artificial intelligence in medicine in Europe. pp. 63–66. Springer (2001)
21. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
22. Lin, W.C., Tsai, C.F., Hu, Y.H., Jhang, J.S.: Clustering-based undersampling in class-imbalanced data. Information Sciences **409**, 17–26 (2017)
23. Lundberg, S., Lee, S.I.: A unified approach to interpreting model predictions (2017)

24. Montavon, G., Binder, A., Lapuschkin, S., Samek, W., Müller, K.R.: Layer-wise relevance propagation: an overview. Explainable AI: interpreting, explaining and visualizing deep learning pp. 193–209 (2019)
25. Olvera-López, J.A., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.: A new fast prototype selection method based on clustering. Pattern Analysis and Applications **13**(2), 131–141 (2010)
26. Ougiaroglou, S., Evangelidis, G.: Efficient dataset size reduction by finding homogeneous clusters. In: Proceedings of the Fifth Balkan Conference in Informatics. pp. 168–173 (2012)
27. Ribeiro, M.T., Singh, S., Guestrin, C.: " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)
28. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
29. Weiss, G.M., Provost, F.: The effect of class distribution on classifier learning: an empirical study. Tech. rep., Rutgers University (2001)
30. Wilson, D.R., Martinez, T.R.: Improved heterogeneous distance functions. Journal of artificial intelligence research **6**, 1–34 (1997)
31. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Machine learning **38**(3), 257–286 (2000)
32. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
33. Yen, S.J., Lee, Y.S.: Cluster-based under-sampling approaches for imbalanced data distributions. Expert Systems with Applications **36**(3), 5718–5727 (2009)