Grad Centroid Activation Mapping for Convolutional Neural Networks

Baptiste Lafabregue[1,2], Jonathan Weber[1], Pierre Gançarski[2], and Germain Forestier[1]
[1]IRIMAS
Université de Haute-Alsace, Mulhouse, France
Email: firstname.lastname@uha.fr
[2]ICube
Université de Strasbourg, Strasbourg, France
Email: lastname@unistra.fr

***Abstract*—An important research effort has been recently dedicated to understand the decision mechanism of deep neural networks. Among them, Class Activation Mapping (CAM) and its variations have proved their capacity to obtain useful insights about Convolutional Neural Network (CNN) models' decisions. However, these methods remain limited to the supervised case regardless of CNN-based advances in unsupervised tasks such as clustering. To fill this gap, we propose a new method called Grad-CeAM for centroid-based clustering methods used on CNN representation. Through an experimental study, we show that our method has the capacity to localize discriminating features used by a CNN model to create its representation and that it can be used to explain the clusters assignment. We also show that this method can be used in different application domains by providing uses cases on time series and images clustering.**

***Keywords*-Activation Map, Clustering, Deep Learning, Interpretability, Time series, Image**

I. Introduction

In the past few years, the development of clustering methods based on Deep Neural Networks (DNNs) have resulted in significant improvements, especially in the image domain [1]–[3]. Among them, models based on Convolutional Neural Networks (CNNs) achieved an unprecedented breakthrough in supervised classification [4]–[6]. This performance of CNNs was also confirmed more recently for clustering, where CNNs have obtained significantly better results than other architectures [1], [7]–[10] making them a major tool for unsupervised approaches. However, despite the high gain in performance, CNNs, like other DNNs, lack of interpretability as the produced models cannot be intuitively understood by a human [11]. Therefore it is difficult, if not impossible, to assert which features are the basis of the model's decision.

Interpretability of a model is a major issue on different levels. When the model fails to obtain the expected predictions, we want to be able to identify major limitations to improve the existing model. Alternatively, when a model reaches high performance, we want to verify that the decision is based on the proper features. This results in better reliability and trust by the users, but it also allows them to learn information from it (e.g. significant patterns, object similarities, etc.). Multiple tools have been already proposed in the literature. This goes from reduction dimension techniques, to better visualize the capacity of the DNNs to separate each class or cluster (e.g. t-SNE [12] or UMAP [13]), to CNNs' activation maps that consist of each layers' filters outputs. For the latter, the information remains limited as it displays the result of all filters but it says little on which element was used in the decision process. Class Activation Mapping (CAM) was proposed to answer to this limitation by computing a heatmap based on the degree of activation of layers' outputs for a given predicted class. First proposed by [14], this version is only applicable to CNN models which do not contain any fully-connected layers and required a specific architecture, or a modification of it with supplementary training. [15] proposed another version, grad-CAM, that could be used without any supplementary process and that generalizes to all CNNs models.

However, both CAM and grad-CAM rely on the softmax layers used for the class prediction to compute the heatmap for each class. Therefore it cannot be applied directly to CNNs models dedicated to clustering. Indeed, clustering models generally rely on a simple encoder, often trained with a decoder on a reconstruction task. Some methods use a clustering layer [2], [3], [8], but, with the exception of a few methods [1], it cannot be directly considered as the equivalent of a softmax layer.

This paper aims to answer to this issue through the following contributions:

- We propose a new method, Gradient-based Centroid Activation Mapping (Grad-CeAM), that extends grad-CAM to CNNs models dedicated to clustering;
- We evaluate our method by showing that it can highlight discriminating features between different clusters and also that it can be used to detect failures in the clustering model;
- We show that our method can be used on standard image datasets but we also that it can be easily applied to other types of data such as time series;
- We provide the code to ensure the reproducibility and reusability of the method on github [1].

The rest of this paper is organized as follows. We first

This work was supported by the ANR TIMES project (grant ANR-17-CE23-0015) of the French Agence Nationale de la Recherche.

[1]https://github.com/blafabregue/gradCeAM

 DOI: https://doi.org/10.1109/ICTAI52525.2021.00033

discuss related works and give more insights on the use of activation maps and clustering CNN methods in Sec. II. In Sec. III, we present our method. In Sec. IV we present the tools used for the evaluation of our method and the results obtained in Sec. V, on its capacity to localize (Sec. V-A) and discriminate (Sec. V-B) clusters' features. Then, we conclude by a discussion on the current limitation of method and perspective in Sec. VI.

II. Background knowledge and related works

In this section, we will first present the CAM and Grad-CAM approaches and, in a second time, the existing propositions to apply them in a clustering context. However, before going further, we want to mention that we will take the image domain as an application example, but similar reasoning can be done for other types of data where CNN models are used, such as time series.

A. CAM and Grad-CAM

The vast majority of clustering methods involving deep learning rely on training an encoder, noted $f()$, to transform the raw data into a latent space more suitable for the clustering task.

Most of the time, the latent representation is a one-dimensional vector that gets rid of specific dimensions of the input. For a set of input images $X = \{x_1, ..., x_N\}$, with $x_i \in \mathbb{R}^{w \times h \times c}$, where w is the width, h the height, and c the number of channels in the image, this will result in its latent representation $Z = \{z_1, ..., z_N\} = \{f(x_1), ..., f(x_N)\}$, with $z_i \in \mathbb{R}^m$, where m is the number of feature of the latent space. This may be an advantage in multiple tasks as we want to be shift-invariant or scale-invariant, since the latent dimension loses the link with the spatial localization of each latent feature in the inputted image.

The encoder is composed of a set of layers of different types. Among them, convolutional layers preserve the dimensional nature of the data (i.e. spatial for images and temporal for time series). Convolutional layers consist of a set of filters. The output is obtained by convolving the filter over the input to obtain a set of activation maps that will be fed to the next layer. For a convolutional layer l, the output of the encoder after the layer l can be seen as a new image, noted $A \in \mathbb{R}^{w' \times h' \times f}$, where w' is the width, h' the height, and f the number of channels/filters of the input. Hence, as the convolution is a local operation, there is a spatial correspondence between the original input and the layer output. Consequently, we can use it to localize the activation degree in the previous layer output but also in the original inputted image.

a) Class Activation Mapping (CAM):: The idea behind this method is to find the linear combination of generated activation maps, A, that resulted in the model prediction of a class c. Their method is straightforward but requires that the network is composed of convolutional layers followed by a Global Average Pooling (GAP) layer and a softmax layer, that we will note S. Thus, the softmax layer's output for the class c, S^c is a linear combination of the GAP output, which is, by construction, the average activation of each filter of the last convolutional layer A, hence :

$$S^c = \sum_{k=1}^{f} W_k * \sum_{i=0}^{w'} \sum_{j=0}^{h'} A_{i,j}^k = \sum_{k=1}^{f} W_{c,k}, \tag{1}$$

where f is the number of filters of layer A and A^k is the k^{th} filter of A, and W_k is the weight of the softmax layer. The CAM is then computed as :

$$CAM = \sum_{k=1}^{f} A^k * W_{c,k} \tag{2}$$

b) grad-CAM:: In this approach, the heatmap is also computed with respect to a class, c. It uses the c^{th} value of the softmax layer, S^c, to compute the gradient value at the last convolutional layer's feature map A. Note that in this case this layer may not necessarily be followed only by the combination of a Global Average Pooling and a softmax layer. Each filter weight is computed for each A's weights $W_{c,k}$ and is global-average-pooled:

$$W_{c,k} = \frac{1}{D} \sum_{i=0}^{w'} \sum_{j=0}^{h'} \frac{\partial S^c}{\partial A_{i,j}^k} \tag{3}$$

where $D = h' \times w'$ is the input dimension. Note that depending on the inputted data it may have more or only one dimension (e.g. the time dimension for time series), but the equation is given for 2D-convolutions.
We obtain from this weights the degree of activation for each layer's filter. The heatmap consists of the weighted sum of all the filter's output with their degree of activation:

$$L_{grad-CAM}^c = ReLU(\sum_k W_{c,k} A^k) \tag{4}$$

Thus, this emphasises the convolution filters that are both highly activated by the input and that contribute to the computation of the gradient. Note that the $ReLU$ function is added to only keep positive contributions to S^c. Therefore by restricting the gradient computation to S^c we only select the convolution results that contribute to this specific class.

B. Clustering application

The adaptation of CAM to clustering methods remains highly limited. This is explained by the absence of a softmax layer in these methods. Therefore, it is difficult to find a way of computing the contribution of each filter to the cluster choice separately.

[16] proposed the first adaptation of CAM for clustering, called CLustering Activation Mapping (CLAM). They used a variation of grad-CAM, Score-CAM [17], that uses a further forward pass with a masked input based on each filter of the last convolutional layer A. It allows to better fix the weight of each filter in the decision based on the softmax output of the evaluated class. However, to apply it to a clustering task, [16] solved the issue by using a framework based on Deep Embedded Clustering (DEC) [3]. DEC framework consists in

pretraining a DNNs encoder and obtaining a initial clustering with K-Means method into K clusters. Then, a clustering layer is added to the encoder, composed of a fully connected layer of size K (the number of clusters), noted q. These layer weights are initialized with centroids values. Therefore, for an input i, the output of the clustering layer at index k, with $1 \leq k \leq K$, $q_{i,k}$ can be seen as the likelihood that it belongs to cluster k. Based on the similarity of the clustering layer and a softmax layer, they just transposed the Score-CAM method by replacing the S^c by q_k.

However, this approach still requires to have a clustering DNN based on the DEC method or that includes a layer with a purpose similar to a softmax layer. Even if such layers are often used, this is not always the case. Several methods are still proposed that only use a simple autoencoder [18].

III. Grad-CeAM: an adaptation of grad-CAM to centroid based clustering

In the clustering case, we do not have a softmax layer with a neuron matching each class as in the classification case. Thus, we want to replace the use of softmax class value S^c with a value that will help to select convolutions that contribute to making the input belong to the analysed cluster. However, instead of classes we have clusters that are often represented by their centroids. The proposed method focuses on clustering methods that involve centroids (e.g. K-Means, EM, DEC, etc.) on top of the CNN's learned representation. Hence, the derivative used to compute the gradient should be done on a loss that is higher when the input is close to the analysed centroid but also lower for other centroids. To do so, we identify which features of the latent representation z contribute the most to obtain the weights $weight_{z,c}$ with respect to the cluster c. Then, we compute a weighted value of the input's latent representation. The weights $weight_{z,c}$ need to emphasize features that contribute the most to making z close to the centroid μ^c of cluster c. To do so, the weights have to:

1) discriminate between all clusters. For example, some of the latent space features may be low or high for all clusters and therefore do not contribute a lot in the cluster choice. This is computed independently of z by the following formula:

$$centroids\_weights^c = \left\| \sum_{\mu^k \in M \setminus \{\mu^c\}} |\mu^k - \mu^c| \right\|_{0-1}, \tag{5}$$

where M is the set of all centroids, $\|.\|_{p-q}$ is the normalization between p and q, and $|.|$ is the absolute value.

2) contribute to making z closer to μ^c than the other centroids. Indeed, we only want features that contribute positively to making z assigned to cluster c. This is done based on the representation z by the following formula:

$$representation\_weights^c(z) = \| |\mu^c - z| \|_{min_M - max_M}, \tag{6}$$

where $min_M = reduce\_min(\sum_{\mu^k \in M \setminus \{\mu^c\}} |\mu^k - z|)$
and $max_M = reduce\_max(\sum_{\mu^k \in M \setminus \{\mu^c\}} |\mu^k - z|)$.

From these two elements, we obtain $weight_{z,c}$ and the final loss:

$$weight_{z,c} = representation\_weights^c(z) * centroids\_weights^c, \tag{7}$$

$$L^c_{centroid}(z) = z * weight_{z,c}, \tag{8}$$

The activation maps are then obtained by:

$$W^c_k = \frac{1}{D} \sum_{i=0}^{w'} \sum_{j=0}^{h'} \frac{\partial L^c_{centroid}(z)}{\partial A^k} \tag{9}$$

The heatmap is then computed as:

$$L^c_{grad-CeAM} = \sum_k ReLU(W^c_k) A^k, \tag{10}$$

For classical grad-CAM, ReLU function is also applied to A^k, but in our case, some negative/masking values may still contribute to the centroid choice. The overall model is summarized in Fig. 1. In the following, we evaluate our method on two aspects: the localization and the discriminability.

IV. Experimental setup

A. Datasets

To validate our method we decided to use both image and time series datasets to show that our method is not restricted to one type of data.

For the image domain we selected:

- MNIST: A dataset of 28-by-28 grayscale images. It consists of 70000 handwritten digits, 10000 in the train set, and 60000 in the test set equally distributed. The digits are centered and size-normalized [19]. Examples of these classes are presented in Fig. 2.
- STL-10: A dataset of 96-by-96 color images. There are 10 classes, with 13000 examples, 5000 in the train set, and 8000 in the test set equally distributed. It consists in classifying the images of the following elements, airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck [20]. Examples of these classes are presented in Fig. 3.

And for the time series domain:

- Coffee: A dataset of univariate time series of length 286. It consists of 56 food spectrographs, 28 in the train set and 28 in the test set equally distributed. It is composed of two classes, each corresponding to a type of coffee beans [21]. Examples of the two classes are presented in Fig. 4.
- Trace: A dataset of univariate time series of length 286. It is a 4 classes synthetic dataset designed to simulate

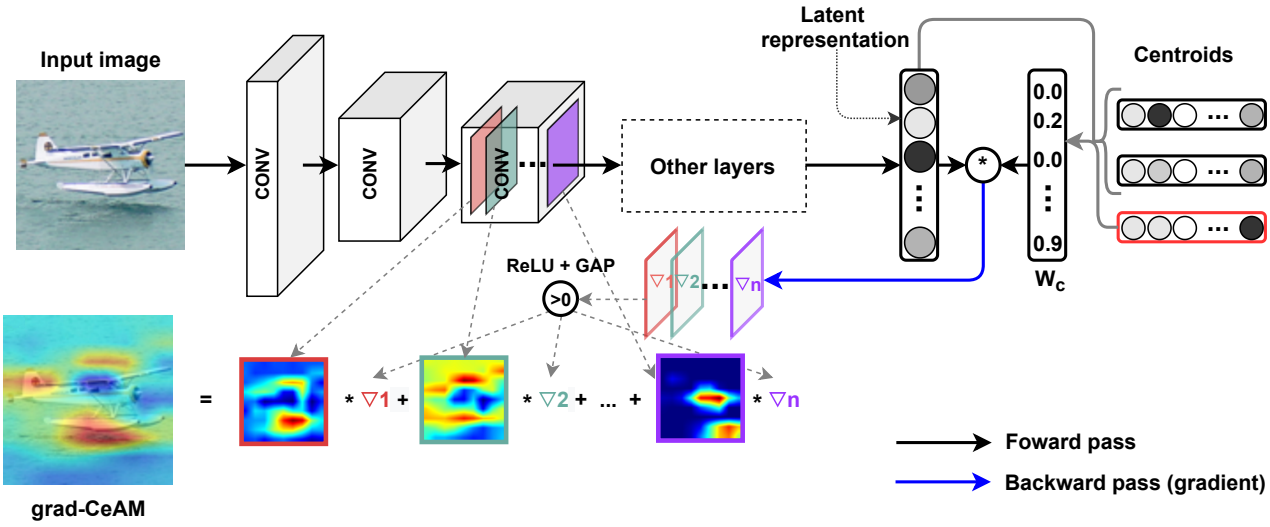
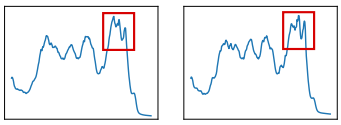


Fig. 1. Grad-CeAM model that combines the last convolutional layer output with the thresholded backpropagated gradient

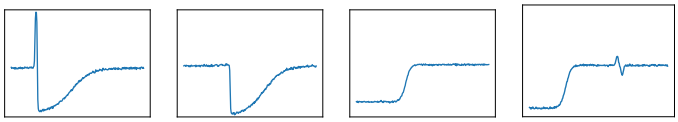
Fig. 2. Examples of MNIST dataset classes

Fig. 3. Examples of STL-10 dataset classes



(a) class 1 (b) class 2

Fig. 4. Examples of Coffee dataset classes, the red squares display the area of decision between the two classes



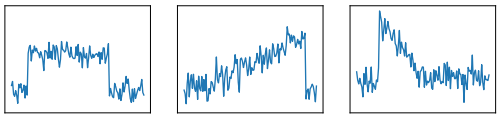
(a) class 1 (b) class 2 (c) class 3 (d) class 4

Fig. 5. Examples of Trace dataset classes

instrumentation failures in a nuclear power plant that are differentiated by their temporal patterns. [22] Examples of the four classes are presented in Fig. 5.

- CBF: A dataset of univariate time series of length 128. It is a 3 classes synthetic dataset designed to discriminate between three shapes, Cylinder (class 1), Bell (class 2), and Funnel (class 3) [23]. Examples of the three classes are presented in Fig. 6.

The three time series datasets are extracted from the UCR archive [24][2].



(a) class 1 (b) class 2 (c) class 3

Fig. 6. Examples of CBF dataset classes

B. Convolutional Neural Networks models

To learn the latent representation we used Convolutional AutoEncoders (CAEs).

For the image datasets, we used a simple encoder-decoder architecture. The encoder is composed of d 2D-convolutional layers, followed by a Global Average Pooling layer (GAP) and a dense layer of size 10 for the embedding layer. The decoder is constructed as a mirror of the d 2D-convolutional layers and ended with a final convolutional layer to fit the input dimension. For STL-10 we used $d = 3$ with parameters set at [32, 64, 64] for filters, with [3, 5, 5] for kernel sizes, and [2, 2, 2] for strides. For MNIST we used $d = 2$ with parameters set at [32, 64] for filters, with [3, 5] for kernel sizes, and [2, 2] for strides.

For the time series datasets, we also used an encoder-decoder architecture. The encoder is based on the ResNet architecture proposed in [25]. The encoder is composed of three residual blocks followed by a GAP and the embedding layer as a dense layer of size 10. Each residual block is first composed of three convolutional layers with a fixed filter size of 64. The filter's length is set to 8, 5, and 3 respectively for the first, second, and third convolution. A ReLU activation function, preceded by a batch normalization operation, is then added at the end of the block. The decoder is constructed as

[2]https://timeseriesclassification.com/

a mirror of the three residual blocks and ended with a final convolutional layer to fit the input dimension.

All CAEs are trained with a classical reconstruction loss:

$$L_r = \frac{1}{n}\sum_{i=1}^{n}(x_i - g(f(x_i)))^2, \tag{11}$$

where $f()$ is the encoder and $g()$ the decoder.

The centroids are computed with the K-Means method executed on the latent representation outputted by the encoder.

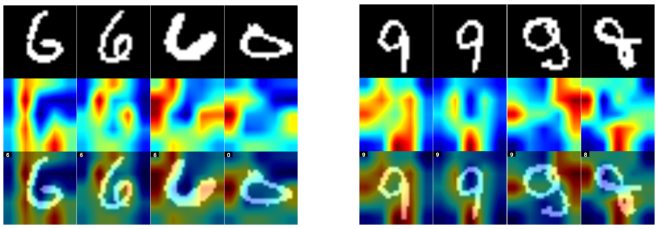
Before going into the evaluation, it should be mentioned that for all the experiments we want to evaluate the interpretability and correctness of the results provided by our visualization method. Thus, whether the actual CNN model obtains good or bad clustering performance, we want that to be reflected in the visualization.

V. Evaluation

A. Localization and model analysis

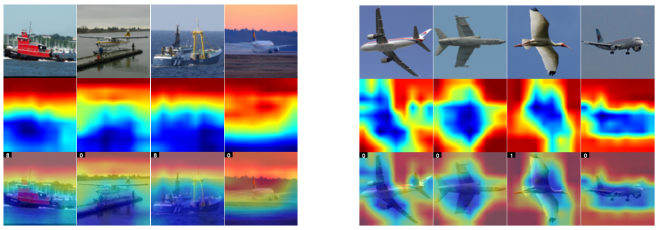
The localization denotes the capacity of the method to correctly identify the features and regions in the input that were used in the decision process. For supervised methods, it is usual to use bounding boxes evaluation on network trained for classification to evaluate if the localization, obtained with CAM tools, manages to highlight the proper area in the image. However, this implies having the data labels and the matching bounding boxes, and furthermore, that the CNN model makes coherent decision. This is not realistic in an unsupervised setup, where the datasets used are often simplified with only one centered element per image or one general pattern for time series, making irrelevant the use of bounding boxes. Moreover, it also assumes that the algorithm bases its decision on the proper part of the input, but given the results obtain by clustering methods we can assume that it is not necessarily the case. To summarize, we do not want to evaluate the capacity of our method to highlight the expected area of interest but the one used by the CNN model. Consequently, we want that the localization proposed by our method matches the frequent patterns among the objects grouped in the same cluster. This will also evaluate the quality of insights outputted by our method.

For the MNIST dataset (Fig. 7), we took two examples, one of the cluster matching the class 6 and one matching class 9. For class 6, the Grad-CeAM is actually showing a high activation on vertical bar of digits 6 and also on the bottom right of the loop. Therefore the assignment of the last sample, which should be grouped with other digits 0, can be better understood as it is not completely rounded on the left side and has a narrow shape on the right side. For class 9, we can observe a symmetric activation (i.e. on the vertical bar and left of the loop). The two most left figures are actually close to the centroid (so more representative) than the two others. For the third sample, it can be observed that the bottom of the digit 9 is not highlighted as it does not match the straight bar from other samples. For the last sample, which is a digit 8, the activation localization is similar to the two first samples,



(a) cluster 6 (b) cluster 9

Fig. 7. Grad-CeAMs heatmap on MNIST dataset. Fig. 7a contains samples from the cluster matching the class 6, and Fig. 7b the one matching the class 9. Red regions correspond to high contribution and blue to almost no contribution to matching the centroid.



(a) cluster 6 (b) cluster 9

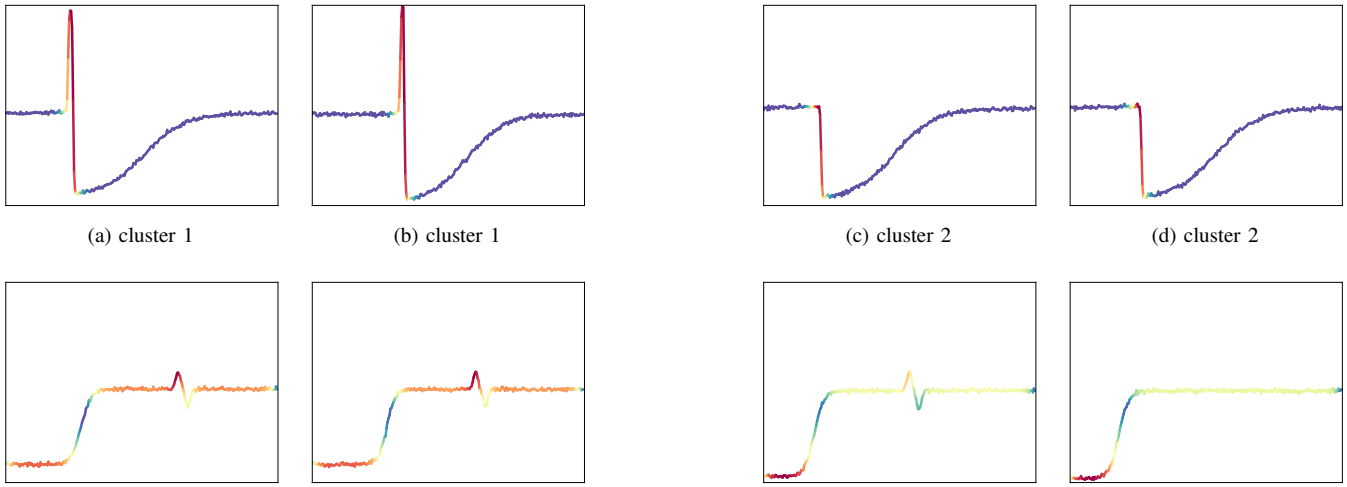
Fig. 8. Grad-CeAM heatmap on STL-10 dataset. Fig. 8a contains samples from the cluster mostly matching the class *boat*, and Fig. 8b the one mostly matching the class *plane*. Red regions correspond to high contribution and blue to almost no contribution to matching the centroid.

as the bottom loop of digit 8 is confused with the bottom bar of digit 9 without taking into consideration the bottom hole. This partially show that the model learned is still sensible to a small rotation.

For the STL-10 dataset, we displayed two clusters' examples in Fig. 8. It can be observed on both datasets that the assignment is actually based on the general image's layout and not on the object itself. The cluster 6 is grouping images where the horizon is dividing the image into two parts and the cluster 9 groups objects centred in a monochrome background (i.e. the sky). The Grad-CeAMs clearly validate this interpretation. For this dataset, the interpretation of the results are not straightforward. However, the Grad-CeAM heatamps on STL-10 clearly show that the network does not focus on the objects themselves, especially for the cluster 6, showing that the features learned by the network are not meaningful for this clustering task.

For the Trace dataset, in Fig. 9, the clusters 1 and 2 are perfectly matching the classes 1 and 2 and Grad-CeAM precisely highlights the two different time patterns. However, there is a confusion of the classes 3 and 4. On this two classes, the clustering seems based on whether the jump happens early or late in the time series and not on the presence of the small variation at the top plateau. This is validated by the Grad-CeAM where the small variation is not highlighted.

For the Coffee dataset, as we have only two classes and we obtain good results (0.82 of NMI score), there is little to conclude by only comparing the two clusters. Therefore, we compare the Grad-CeAM localization with a CAEs with a



(a) cluster 1 (b) cluster 1 (c) cluster 2 (d) cluster 2

(e) cluster 3 (f) cluster 3 (g) cluster 4 (h) cluster 4

Fig. 9. Grad-CeAM heatmap on Trace dataset. Two samples are displayed per cluster. Red regions correspond to high contribution and blue to almost no contribution to matching the centroid (colors are smoothed for visual clarity and better reflect filters' size).

lower clustering performance, measured by their Normalized Mutual Information (NMI) score. The *low* model is obtained by selecting the worst model among multiple executions. In Fig. 10, we can easily identify the difference between the two models. The *high* model catches the area of discrimination, whereas the *low* model focus on the end of the time series (low or high bump) that has no involvement in the class discrimination.

For the CBF dataset, in Fig. 11, we displayed the comparison of heatmap generated from the three centroids on the same time series to show the differences between the three centroids' Grad-CeAMs. For the Grad-CeAMs of the cluster 1 (Fig. 11a to 11c), the activation is strong when the signal is stable, and especially low when there is an increase in the signal. At the opposite, the Grad-CeAMs of the cluster 3 are clearly focusing on strong increase of the signal. It should be noted that the heatmaps' colors are normalized, hence the highly red part in Fig.11c is just telling that this part of the time series contributes more to the decision, but overall the degree of activation/contribution is lower than in the red part of the Fig. 11a.

The results obtained on these 5 datasets illustrate that our method localization help to understand the clustering assignment and therefore seems coherent with the obtained clustering.

B. Cluster discrimination

In this section, we want to evaluate if the features highlighted by Grad-CeAM are specific to each cluster and allow to understand the choice between clusters.

If the previous assertion is true, amplifying (or reduce) the values of the selected feature should respectively have a positive (or negative) effect on the choice of the analyzed cluster. To evaluate this, we have to use the gradient computed for Grad-CeAM to weight the output of the last convolutional layer A. The new output is computed as:

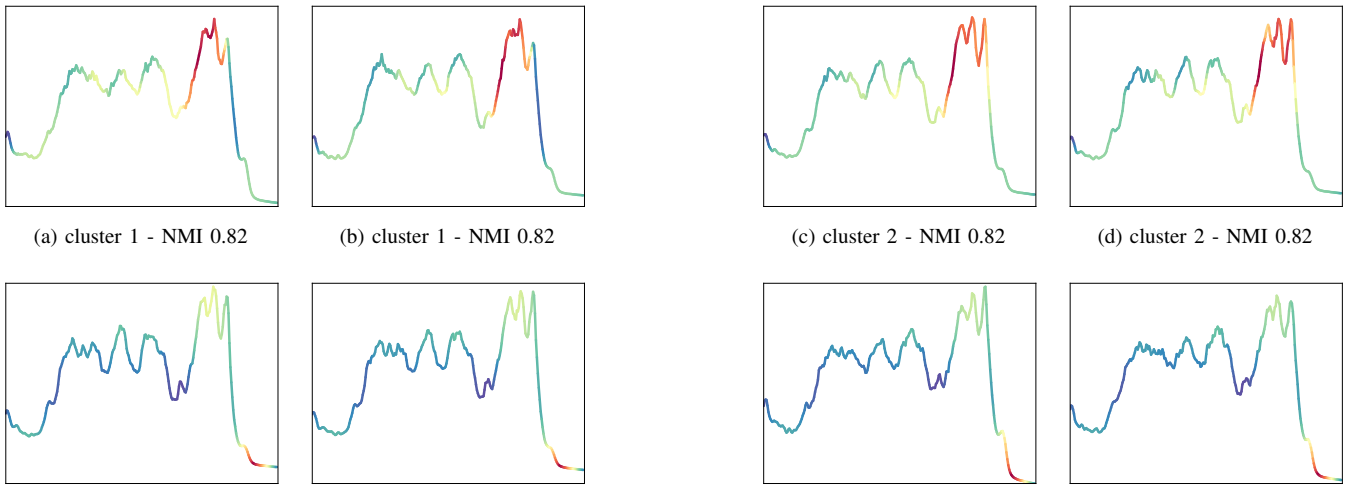
$$A_{c-graded} = \|ReLU(a_k^c)\|_{0-2}\, A^k \tag{12}$$

The a_k^c weights are normalized between 0 and 2 to amplify the high values. Then, $A_{c-graded}$ is used as input of the CAE's next layer to obtain the modified representation $Z^c = \{z_1^c, ..., z_N^c\}$. Finally, a new clustering is computed on Z^c but based on the original centroids, M, used to compute the Grad-CeAM model. Thus, by measuring the proportion of objects assigned to the cluster c, we can asses if the modification added improved the likelihood of an input to be assigned to the cluster c.

TABLE I
COMPARISON OF CLUSTER CARDINALITY BEFORE AND AFTER APPLYING GRADIENTS WEIGHTS TO THE LAST CONVOLUTIONAL LAYER.

Dataset	STL-10	MNIST	Trace	Coffee	CBF
Median	1.1	1.4	1.5	1.2	1.7
Average	1.2	1.4	1.7	1.2	1.4
Minimum	0.0	0.5	1.0	0.0	0.8
Maximum	3.1	5.8	2.8	2.3	1.8

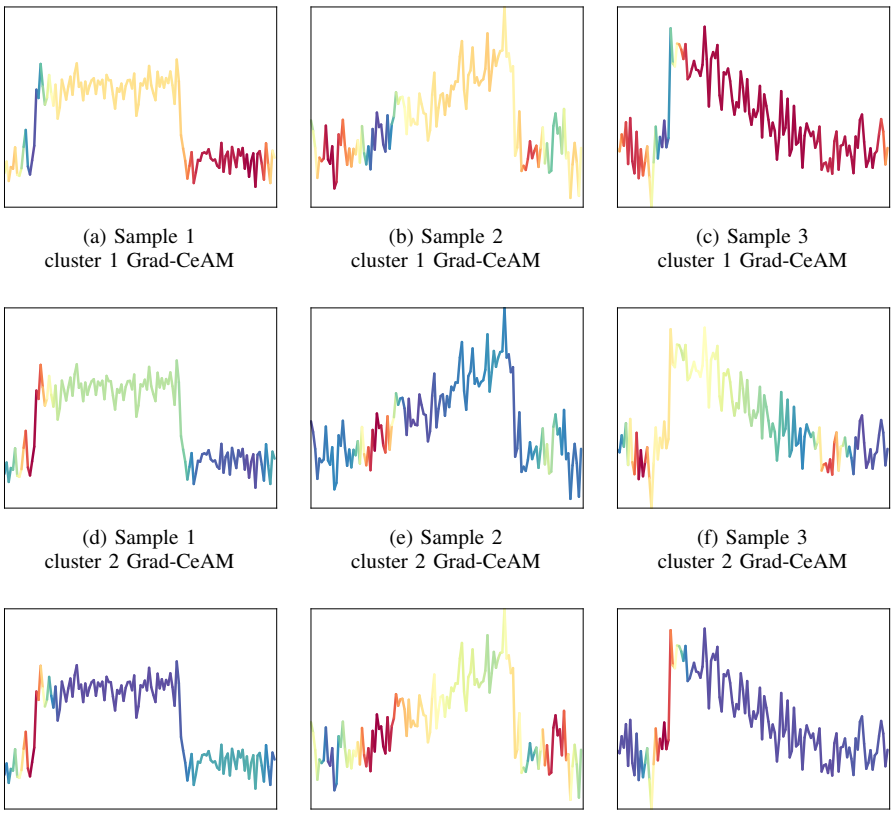
In Tab. I, we have reported the median, average, minimal, and maximal proportion change of cluster assignations. It can be observed that the cluster used to set the weights largely benefits from the amplification. Hence, this shows that, on average, Grad-CeAM selects the correct features for each centroid. However, we can also point that, even though the median gain is positive (1.3 in average, i.e. 30% cardinality gain), some clusters do not benefit from the modification. For all dataset, except Trace, there is at least one cluster that loses in cardinality or even disappears (i.e. no object is assigned to this cluster). For MNIST and CBF, this seems marginal, as it is limited to one cluster, all the others having at least a stable cardinality. For Coffee, one cluster absorbs the other



(a) cluster 1 - NMI 0.82 (b) cluster 1 - NMI 0.82 (c) cluster 2 - NMI 0.82 (d) cluster 2 - NMI 0.82

(e) cluster 1 - NMI 0.03 (f) cluster 1 - NMI 0.03 (g) cluster 2 - NMI 0.03 (h) cluster 2 - NMI 0.03

Fig. 10. Grad-CeAM heatmap on Coffee dataset, where 10a,10b,10c and 10d are samples of each cluster from clustering with 0.82 of NMI score and 10e,10f, 10g and 10h are samples of each cluster with a representation with 0.03 of NMI score. Red regions correspond to high contribution and blue to almost no contribution to matching the centroid (colors are smoothed for visual clarity and better reflect filters' size).



(a) Sample 1 cluster 1 Grad-CeAM (b) Sample 2 cluster 1 Grad-CeAM (c) Sample 3 cluster 1 Grad-CeAM

(d) Sample 1 cluster 2 Grad-CeAM (e) Sample 2 cluster 2 Grad-CeAM (f) Sample 3 cluster 2 Grad-CeAM

(g) Sample 1 cluster 3 Grad-CeAM (h) Sample 2 cluster 3 Grad-CeAM (i) Sample 3 cluster 3 Grad-CeAM

Fig. 11. Grad-CeAM heatmap on CBF dataset, where 11a,11b, and 11c are samples from respectively cluster 1, 2 and 3 colorized with the Grad-CeAM of cluster 1's centroid, 11d,11e, and 11f are the same samples but with the Grad-CeAM of cluster 2's centroid colorization, and 11g,11h, and 11i for cluster 3. Red regions correspond to high contribution and blue to almost no contribution to matching the centroid (colors are smoothed for visual clarity and better reflect filters' size).

regardless of the centroid used for the Grad-CeAM. This is explained by one cluster being defined by a lower value on some discriminating features. Therefore, when we amplify the feature its value increases instead of being lowered (to be closer to the centroid), resulting in favoring the other cluster. These low values usually mean a weak activation from the CNN layer coming from either a small signal amplitude or its absence, e.g. a smaller spike for the Coffee dataset. For STL-10, the explanation is more difficult to find from observing the learned representation. However, this may be partially explained by two elements. First, similar to the Coffee dataset we have a cluster defined by lower values on some features. Secondly, the low results of the CAEs representation (only 0.2 of NMI score) may suppose that centroids are not well defined or that they rely on degenerated features. A similar behavior was also observed for MNIST dataset when the NMI score was too low. Nevertheless, this shows that our method has difficulty identifying all elements used by the CNN model to generate the prediction, especially to translate the absence of a specific pattern (i.e. low activation map's values).

VI. Conclusion

In this work, we proposed a new method, Grad-Centroid Activation Mapping (Grad-CeAM), to visualize activation localization for any CNN-based model in the context of clustering for centroid-based algorithms. We state that our method can provide a good representation of the importance of each activation map. We support this statement experimentally by showing that our method outputs relevant activation heatmaps that are both localized discriminating between each cluster. We also show that our method can provide useful insight on the clustering assignment that can help the researcher to analyze their model. Finally, even if we obtain good overall results, we also show that our method struggles to completely catch the characteristics discriminating features for some centroids, especially if they result from a weak activation in the CNN model.

Acknowledgment

This work was supported by the ANR TIMES project (grant ANR-17-CE23-0015) of the French Agence Nationale de la Recherche.

References

[1] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5736–5745.

[2] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, and I. Craddock, "N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding," *arXiv preprint arXiv:1908.05968*, 2019.

[3] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[4] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE international symposium on circuits and systems*. IEEE, 2010, pp. 253–256.

[7] B. D. de Vos, F. F. Berendsen, M. A. Viergever, M. Staring, and I. Išgum, "End-to-end unsupervised deformable image registration with a convolutional neural network," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 204–212.

[8] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *International conference on neural information processing*. Springer, 2017, pp. 373–382.

[9] B. Lafabregue, J. Weber, P. Gançarski, and G. Forestier, "Deep constrained clustering applied to satellite image time series," in *ECML/PKDD Workshop on Machine Learning for Earth Observation Data (MACLEAN)*, 2019.

[10] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "Clustergan: Latent space clustering in generative adversarial networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4610–4617.

[11] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[12] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[13] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[14] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[15] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[16] S. Ryan, N. Carlson, H. Butler, T. Fingerlin, L. Maier, and F. Xing, "Cluster activation mapping with applications to medical imaging," *arXiv preprint arXiv:2010.04794*, 2020.

[17] H. Wang, M. Du, F. Yang, and Z. Zhang, "Score-cam: Improved visual explanations via score-weighted class activation mapping," *arXiv preprint arXiv:1910.01279*, 2019.

[18] T. A. Geddes, T. Kim, L. Nan, J. G. Burchfield, J. Y. Yang, D. Tao, and P. Yang, "Autoencoder-based cluster ensembles for single-cell rna-seq data analysis," *BMC bioinformatics*, vol. 20, no. 19, pp. 1–11, 2019.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[21] R. Briandet, E. K. Kemsley, and R. H. Wilson, "Discrimination of arabica and robusta in instant coffee by fourier transform infrared spectroscopy and chemometrics," *Journal of agricultural and food chemistry*, vol. 44, no. 1, pp. 170–174, 1996.

[22] D. Roverso, "Multivariate temporal classification by windowed wavelet decomposition and recurrent neural networks," in *3rd ANS international topical meeting on nuclear plant instrumentation, control and human-machine interface*, vol. 20. Citeseer, 2000.

[23] N. Saito, "Local feature extraction and its applications using a library of bases," Ph.D. dissertation, Yale University, 1994.

[24] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[25] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1578–1585.