Machine Learning
https://doi.org/10.1007/s10994-024-06529-8

Explaining Siamese networks in few-shot learning

Andrea Fedele[1,2] · **Riccardo Guidotti**[1,2] · **Dino Pedreschi**[1,2]

Received: 2 March 2023 / Revised: 29 January 2024 / Accepted: 14 February 2024

Abstract
Machine learning models often struggle to generalize accurately when tested on new class distributions that were not present in their training data. This is a significant challenge for real-world applications that require quick adaptation without the need for retraining. To address this issue, few-shot learning frameworks, which includes models such as Siamese Networks, have been proposed. Siamese Networks learn similarity between pairs of records through a metric that can be easily extended to new, unseen classes. However, these systems lack interpretability, which can hinder their use in certain applications. To address this, we propose a data-agnostic method to explain the outcomes of Siamese Networks in the context of few-shot learning. Our explanation method is based on a post-hoc perturbation-based procedure that evaluates the contribution of individual input features to the final outcome. As such, it falls under the category of post-hoc explanation methods. We present two variants, one that considers each input feature independently, and another that evaluates the interplay between features. Additionally, we propose two perturbation procedures to evaluate feature contributions. Qualitative and quantitative results demonstrate that our method is able to identify highly discriminant intra-class and inter-class characteristics, as well as predictive behaviors that lead to misclassification by relying on incorrect features.

Keywords Explainable artificial intelligence · Interpretable machine learning · Siamese networks · Few-shot learning · Data-agnostic explanations

Editors: Dino Ienco, Roberto Interdonato, Pascal Poncelet.

✉ Riccardo Guidotti
riccardo.guidotti@unipi.it; riccardo.guidotti@isti.cnr.it

Andrea Fedele
andrea.fedele@phd.unipi.it; andrea.fedele@isti.cnr.it

Dino Pedreschi
dino.pedreschi@unipi.it; dino.pedreschi@isti.cnr.it

[1] University of Pisa, Largo B. Pontecorvo, 3, 56127 Pisa, Italy

[2] ISTI-CNR, Via G. Moruzzi, 1, 56124 Pisa, Italy

Published online: 29 April 2024

1 Introduction

In recent years, Artificial Intelligence (AI) has made significant progress due to the availability of large datasets, powerful computing devices, and the development of sophisticated algorithms (Haenlein & Kaplan, 2019). Machine learning (ML) models are commonly used in AI systems because of their success in various fields such as image processing, time series analysis, and audio signal processing (Sarker, 2021; Purwins et al., 2019). However, traditional ML systems have a major limitation in that they rely on large-scale datasets, while real-world applications often have constraints that result in limited data availability (Ienca & Vayena, 2020; Jiang et al., 2014; Fries et al., 2017). Technical issues may limit the collection of training data, while ethical or privacy concerns may restrict data access (Ienca & Vayena, 2020). Furthermore, traditional ML systems struggle to generalize from few samples and their performance is often better for classes with more training samples and worse for classes with fewer samples (Rahman et al., 2018). As a result, these systems are limited in their ability to expand their knowledge beyond the scope of the data they were trained on. In contrast, humans are able to quickly generalize from prior knowledge. For example, if a child is presented with a few pictures of a person or animal he/she has never seen before, he/she will still be able to match and identify the correct individual among a reasonable number of pictures portraying different subjects/animals.

To overcome these limitations, recent studies proposed the use of *few-shot learning* frameworks, where a ML model must learn to classify new classes with only a limited number of labeled samples per class (Wang, 2020). Few-shot learning methods typically consider a *C-way k-shot* classification task, where C represents the number of classes the model must recognize, while k is the number of labeled samples per class. The set of labeled samples is known as the *support set*, i.e., an auxiliary set of data that serves as guidance for the classifier. The goal of few-shot learning is to predict the class of an instance when only a small number of examples of that specific class are available. In this paper, our focus is on classifying an unseen sample using just one labeled sample of that specific class during inference. This means that the model has had no exposure to the test samples during training, as we have kept the training, validation, and test sets separate. We refer to this task as *C-way* **one**-*shot* learning, signifying that the support set comprises a single sample for each of the C classes. Consequently, the model's objective is to correctly classify an unseen query class with only one additional sample from that class in the support set during inference.

Several algorithms have been proposed to address the challenge few-shot metric-learning including Siamese Networks (Koch et al., 2015), Matching Networks (Vinyals et al., 2016), Relation Networks (Sung et al., 2018), and Prototypical Networks (Snell et al., 2017). These algorithms aim to learn a metric in an embedding space that can then be used to determine the similarity between unseen samples. Siamese Networks (SNs) (Koch et al., 2015) are composed of two or more identical encoding sub-networks that map inputs into an embedding space, where a distance function is applied to calculate the distance between the resulting embedded representations. A similarity score is then computed based on this distance in the latent space. To date, the few-shot learning paradigm has been applied to image processing (Koch et al., 2015; Qiao et al., 2018; Liu et al., 2019), time-series analysis (Iwata & Kumagai, 2020; Gupta et al., 2021), and audio classification (Vélez, 2018; Honka, 2019).

However, one of the main challenges of SNs is the lack of explainability. Indeed, it is challenging to comprehend how these architectures can correctly generalize on unseen

samples. Understanding the reason why a model takes a specific decision is hugely important to developers, organizations, and end-users on which such decision falls upon. While end-users may prioritize understanding an outcome's explanation over the outcome itself, developers can use explanations to identify potential issues with a model and repeat training procedures in a controlled environment. Also, stakeholders may need to comprehend the reasoning behind a model's decisions before deploying it in real-world scenarios that may have a significant impact on people's lives. Indeed, the "right to know" (Dimitrova, 2020), refers to an individual's right to receive an explanation for a specific outcome produced by an algorithm. In recent years, researchers examined the eXplainable Artificial Intelligence (XAI) topic from various perspectives (Guidotti, 2019; Adadi, 2018; Miller, 2019). A characterization of XAI techniques distinguishes between *gradient-based* and *perturbation-based* approaches (Guidotti, 2019; Adadi, 2018). While both aim to understand the contribution that each input feature has on a specific outcome, they solve the problem differently. Gradient-based approaches estimate feature contribution through forward and backward propagation in the network, while perturbation-based methods perturb the input and measure changes in the output relative to the original input. Although many XAI methods received positive feedback from the research community, only a limited number are designed to explain Siamese Networks (SNs) and, to the best of our knowledge, none has been developed to work on different data types within the framework of *C-way one-shot learning*.

In this work, we introduce a local data-agnostic explanation method for SNs in the setting of *C-way one-shot learning*. Our *SIamese Networks EXplainer* (SINEX) aims to uncover the decisive features that enable the model to learn and generalize in a manner that resembles human cognition when making predictions on image, time series, and audio data. We aim to use our explainer to address questions such as: *What factors is the model considering when it accurately identifies the class of two unseen objects? What makes a particular sample more similar to one object than to others? Why is the model miss-classifying a specific object?* SINEX employs a perturbation approach to calculate the contribution of each feature based on a segment-weighted-average evaluation. Additionally, we present a coalition-based variant of SINEX, called SINEXC, which takes into account the interaction between different parts of the input. These contribution values can be visualized as heatmaps, providing an intuitive representation of the behavior of Siamese Networks. An illustrative SINEX explanation of a 5-way 1-shot classification is presented in Fig. 1. In the top left corner, the query sample x to be classified is shown, followed by a support set consisting of 5 samples (labeled from s_1 to s_5). Above each i^{th} support sample, the similarity value computed by the Siamese Network between the pair (x, s_i), represented in the range of [0, 1], is displayed. The bottom row displays the SINEX contribution heatmaps for each i^{th} sample in the support set, denoted as h_1 to h_5. Positive contributions are highlighted in red, while negative contributions are shown in blue. Areas in grey are considered neutral with respect to the similarity score outcome. These heatmaps are min-max normalized within the same tasks and can be compared between them. In this example, both h_1 and h_3 show the largest positively influencing segments overall.

We apply SINEX to explain SNs in the context of *5-way one-shot* learning on grayscale and RGB images, time-series, and audio data. Our results demonstrate the effectiveness of SINEX in identifying positive and negative contributing areas to the network outcome. Furthermore, our approach uncovers limitations in SNs, including erroneous dependence on specific colors (RGB images) or pixels (grayscale images), which can result in classification errors. This paper extends the conference version (Fedele et al., 2022) in several aspects. First, we introduce a perturbation methodology for feature contribution

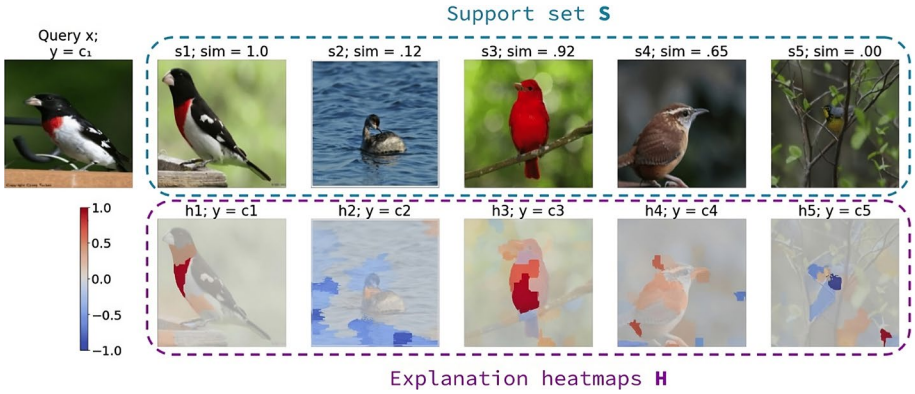

Fig. 1 SINEX explanation on a 5-way 1-shot classification task on the `Caltech-UCSD Birds 200` dataset. Top row: query sample x followed by the support set samples s_1 to s_5 from left to right. The SN similarity computed between each (x, s_i) pair is presented above the corresponding s_i sample. Bottom row: explanation heatmaps h_1 to h_5 corresponding to the s_1 to s_5 support samples. Heatmaps display positive (red), negative (blue), and neutral (gray) contributions to the similarity score. Class labels from c_1 to c_5 are: Rose breasted Grosbeak, Eared Grebe, Summer Tanager, Carolina Wren and Canada Warbler. This layout is repeated as such for all explanations on this paper. Best viewed in colors (Color figure online)

allocation, that computes the importance of specific features by excluding them instead of solely considering them. Second, we generalize the applicability of SINEX to any data type by extending the input data handled, including images and time series besides audio data. Third, we extend the explainer's compatibility with the widely used 3-branched Siamese Network architectures (Hoffer, 2015), which, in turn, accomodate for *distance-based* SN approaches, besides the *similarity-based* ones previously addressed. Fourth, we introduce the ϵ-LRP explainability technique as an additional baseline in our performance assessment. Fifth, we investigate the impact of shifted support sets on both the SN performance and the explainer itself. Finally, we perform a *dependence-ness* analysis of positive and negative contribution features by introducing two novel metrics.

The rest of paper is organized as follows. In Sect. 2, we review relevant studies. Section 3 outlines the problem we address, while in Sect. 4 we illustrate our proposed solutions. The results of our experiments are presented in Sect. 5 and extensively discussed in Sect. 6. Finally, Sect. 7 summarizes our contributions and suggests future research avenues.

2 Related works

In this section, we review existing works that aim to explain the behavior of Siamese Networks (SNs). In Zhang (2019) the authors focus on visualizing and *sonifying* the input patterns that activate certain neurons in the network using the activation maximization approach (Erhan et al., 2010). The authors suggest that by visualizing the patterns that activate random neurons from each layer, it is possible to get an idea of what the network considers important. However, the limited consideration of random neurons may result in an isolated effect with respect to the overall function of the SN. Additionally, it is not possible to determine whether the returned features have a positive or negative impact on the outcome. A similar approach is described in Acconcjaioco (2020), where the system is

developed for one-shot learning to identify bird species. The authors visualize how audio spectrograms (Flanagan, 2013) are decomposed by each layer and consider the last one as the explanation layer. Both in Zhang (2019) and Acconcjaioco (2020), the explanation is only derived from the convolutional encoders and does not take into account the core similarity scoring layer of the SN.

In Utkin (2020), the authors use a special auto-encoder as the core of their explanation algorithm, similarly to Guidotti (2019); Looveren (2021); Naudé (2020). The encoder is trained to reconstruct the input instances of the training set using the embedded representation provided by the SN's built-in embedder. The decoder is then trained to reconstruct the original input based on the hidden representation. Once the auto-encoder is trained, it receives a pair of inputs to explain, which are also given to the SN. The vectors produced by the SN's encoders are then perturbed on what the authors refer to as *important features*. These features are chosen based on the smallest distance between the two inputs if they are semantically similar, or the largest distance if they are dissimilar. The perturbed vectors are then passed through the decoder, which maps them back to the original input space. The embedded vectors are randomly perturbed, and the mean contribution value of each feature is calculated as the difference between its value in the reconstructed input after perturbation and its value in the reconstructed input without perturbation. The proposal suffers from various weaknesses like including a large number of parameters, the need for a large amount of data to train the auto-encoder, the access to the training set, and the requirement of training an additional ML model. In addition, also in this case, the SN's distance function and similarity score are bypassed by the explainer.

In Chen (2021), is introduced another post-hoc explanation approach for SNs. The authors highlight the concern that existing perturbation-based XAI methods might become overly sensitive to irrelevant perturbations when dealing with unseen instances in the support set. To address this issue, they find global, invariant salient features for individual object using self-supervision. Then, they formulate an optimization problem to adapt the global salient features to explain a SN prediction for an input pair. The adaptation balances the conformity to the invariance and the local flexibility when comparing a query to different references. The authors design a gradient descent algorithm to solve a constrained optimization problem with KL-divergence regularization. However, also Chen (2021) does not account for the similarity scoring layer and only operates on the embedded representation of the data.

In Ye (2020), a different approach is taken, where a *class-to-class SN* (C2C-SN) is trained to understand both the similarities and differences between classes. The authors show how the C2C-SN can be used for explanation purposes through prototypical case finding and contrasting cases. However, the proposal of Ye (2020) differs from our work as it does not query the model on unseen classes.

A very recent work (Tummala & Suresh, 2023) introduces a novel class activation mapping, which highlights critical regions for classification. The saliency map is crafted by employing the l_1 distance lambda layer as a weight vector, which is then multiplied with the final convolutional layer responsible for query sample embedding.

Finally, in Fiaidhi et al. (2022), the authors exploit SNs to identify the ulcerative colitis from few training samples, while enhancing the network's explainability by combining it with a LSTM model that provides relevant textual captions for a specific outcome. SNs are trained on RGB images using a triplet loss (Schroff et al., 2015) and are asked to classify eight different classes. With the help of expert-provided textual captions, the authors are able to train a LSTM model on textual input and form an explanation that combined the SN prediction with a textual caption learned from a field expert. Differently from the other existing explainers for SNs, this work shows that explanations might come in different

forms and that SNs can be embedded into bigger architectures instead of being treated as standalone systems.

The explanation approach presented in this paper differs from those described in Zhang (2019), Acconcjaioco (2020), and Chen (2021) as they utilize *gradient-based* methods that primarily focus on the encoder component of SNs. These works only examine the last convolutional layer of the network, ignoring the overall architecture of the SN. Moreover, our approach eliminates the need for training additional models and focuses on assessing differences in the SN outcome through input space perturbations, unlike (Utkin, 2020). While SINEX perturbs samples in the input space and measures differences in the SN outcome score, this approach perturbs the latent space post-CNN encoding and generates heatmaps for perturbed samples using a pre-trained decoder. Consequently, we decided not to use it for comparison since our goal is to evaluate the contribution of input features in the input space.

Additionally, our approach differs from Fiaidhi et al. (2022) as we aim to explain the SN as it is, without the need for training samples or external knowledge. Lastly, our explanation method can be directly applied during prediction time. Our aim is to provide a comprehensive explanation of how a SN works, with a particular emphasis on the layer that generates the final similarity or distance score. Additionally, our approach addresses the issue of explaining SNs within the context of C-way one-shot learning, an area that has not been explored in previous works.

3 Problem formulation

In few-shot learning, a dataset $D = \langle X, y \rangle$ is composed of n input samples, $X = \{x_1, \ldots, x_n\}$, and their corresponding class labels, $y = \{y_1, \ldots, y_n\}$. The class labels indicate which y_i class each x_i input sample belongs to, where $y_i \in [0, \ldots, L-1]$. L is the total number of classes, which in few-shot learning is higher than traditional multi-class problems. In our experiments, L ranges from a minimum of 50 to a maximum of 60. In *C-way one-shot learning*, the support set S contains C input samples, each belonging to a different class and only appearing once in S. Typically, C is much smaller than L. In our experiments, C is set to 5. In this framework, a SN is a deep learning model f that takes as input a support (or reference) set $S = \langle \{s_1, \ldots, s_C\}, \{y_1, \ldots, y_C\} \rangle$, a query instance x with an unknown label, and predicts the class label of x by comparing it to each input sample s_i in the support set. Each y_i present in the support set represent the class label of the sample s_i. This comparison can be done using either a similarity *sim* (Koch et al., 2015), or a distance function, *dis* (Hoffer, 2015), to select the highest/lowest similarity/distance score, i.e.,

$$y_i = f(x, S) = \underset{\forall s_i, y_i \in S}{\arg\max}\ \text{sim}(x, s_i) \qquad y_i = f(x, S) = \underset{\forall s_i, y_i \in S}{\arg\min}\ \text{dis}(x, s_i).$$

In this context, a correct classification occurs when the class y_i of the support sample s_i, which yields the highest/lowest similarity/distance score, matches the class of the query input x. Otherwise, an incorrect classification occurs.

In our setting, we consider samples in the dataset X including images, time-series, or audio inputs. An *image* is typically represented as a matrix in $\mathbb{R}^{p \times q \times z}$, where p and q are the width and height Cartesian coordinate of the image and z is the number of color channels (usually 3 for RGB and 1 for grayscale images). Each element $x_{i,j,k,w}$ represents the intensity value at width j, height k and color channel w for the i^{th} image. The intensity is usually

Fig. 2 Left: RGB image of Rose breasted Grosbeak bird. Center: grayscale image representing a time-series formed by taking the height profile of a written word. Right: log-mel spectrogram of an audio recording featuring a speaker saying "zero" out loud. Darker areas represents lower dBs, i.e., silence, while lighter pixels indicate sounds that can be heard by humans (Color figure online)

expressed as a value within the range of [0, 255]. A *time series* is an ordered set of p real-valued observations (or time steps) in $\mathbb{R}^p$. We represent a time series as a grayscale image by plotting it on a Cartesian coordinate system as $x_{i,j,k}$ where j represents the value plotted on the temporal-axis and k represent the measurement value at that timestamp for the i^{th} time series. Finally, an *audio* can be represented as a spectrogram (Acconcjaioco, 2020; Flanagan, 2013), which consist of a matrix in $\mathbb{R}^{p \times q}$, where q is the length of the audio track, and p is the number of observed frequencies. Each element of the matrix $x_{i,j,k}$ represents the intensity value at time j of frequency k for the i^{th} audio track. The intensity is usually expressed in decibel (dB). Figure 2 shows an RGB image, a grayscale representation of a time-series and an audio spectrogram.

The main goal of this paper is to identify what a pre-trained SN f considers among the characteristics of the records in the support set S when assigning class y_i to a query instance x. To do so, we propose a local data-agnostic post-hoc explanation method g, which takes as input f, a set of support samples S, and a query instance x and returns an explanation E. More formally, our objective is to define a function g such that $E = g(f, x, S)$. The explanation E is formalized as a set of heatmaps $E = \{h_1, \ldots, h_C\}$, where each h_i is the heatmap of the support sample $s_i, y_i \in S$ and represents the importance/saliency for each feature of its matrix value. For RGB images, the value $h_{i,j,k,w}$ indicate the importance of the pixel at width j and height k for the w^{th} channel color. The same definition applies for grayscale images, regardless of whether they were generated from a time-series input or not. For audio tracks, the value $h_{i,j,k}$ indicates the importance/saliency of the k^{th} frequency at time j for the i^{th} support set spectrogram.

4 Siamese networks explainer

In this section, we describe SINEX, a local data-agnostic *SIamese Networks EXplainer* for *C-way one-shot learning*. SINEX is our proposal for implementing the function g to explain a SN f w.r.t. a query instance x and a support set S, as described in the previous section. In particular, SINEX unveils the output of the SN by generating an explanation based on the final layers that measure either the similarity or the distance score. We design SINEX as a *perturbation-based* method that measures the prediction of similarity in 2-branched SNs (2SN) or distance in 3-branched SNs (3SN) after various input perturbations and repeated

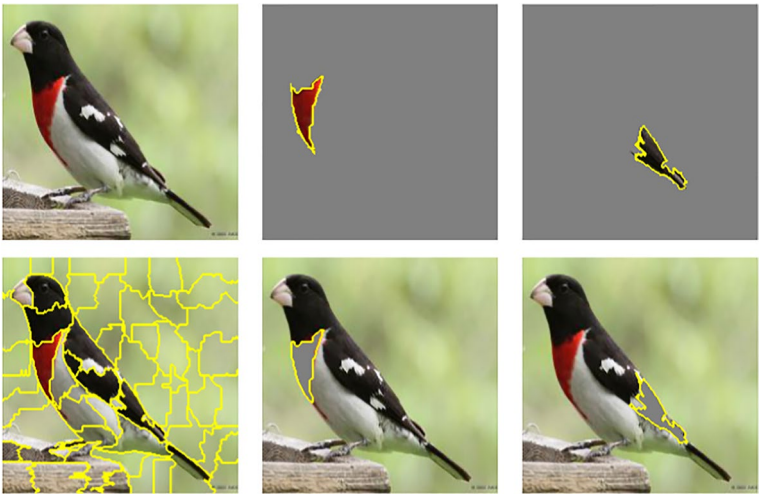

Fig. 3 First column: support set sample on the top row, and its segmentation on 56 regions in the bottom row. First row: $\omega{=}\neg$ perturbation on the 24th and 35th segments respectively. Second row: $\omega{=}1$ perturbation on the same segments. Best viewed in colors (Color figure online)

queries of the SN. In our *C-way one-shot* setting where the query input x is classified based on the instances in the support set S, we choose to segment and perturb the instances in S and observe how the outcome estimation between x and s_i changes when parts of s_i are hidden from the network.

We introduce two perturbation approaches, and use SINEX_{ω} to distinguish them. The first approach with $\omega{=}\neg$, named $\text{SINEX}_{\neg}$, involves measuring the contribution of a specific segment by keeping it active while "silencing" all the other segments, i.e., replacing them with non-informative values. The second approach with $\omega{=}1$, named SINEX_{1}, measures the contribution of a specific segment by "silencing" it, while keeping all other segments active. Examples of both perturbation procedures are illustrated in Fig. 3. In the following, we first elucidate the segmentation process that precedes the perturbation runs. Then, we present SINEX and SINEXC, a coalition-based variant of SINEX that extends the perturbation procedure to multiple segments rather than single ones. After that, we delve into the explanations generated by SINEX and SINEXC, providing details on how to interpret and compare them. Finally, we describe how 2-branched SNs (2SNs) differ from 3-branched SNs (3SNs) and explore how our explainers can provide support for both.[1]

4.1 SINEX segmentation approach

Input matrices, whether representing images, time series, or audio spectrograms, offer various avenues for perturbation. A common approach is the window-occlusion-based technique (Zeiler, 2014). However, this method faces several limitations. Fixed-size occlusion windows could yield inaccurate results, as the contribution of the features can vary significantly depending on the window size. Additionally, determining the correct window size can be difficult, as the same size may produce different outcomes even for different

[1] SINEX_{1} perturbation and 3SNs distance-based support not investigated in Fedele et al. (2022).

instances of the same class. Furthermore, dividing the input into fixed-length windows assumes that the two axes are independent, which is a rare assumption for inputs like images and spectrograms. For example, relying solely on time segmentation for a spectrogram means assuming that every sound event starts and ends at the same moment in all recordings of a given class, which is unrealistic in real-world scenarios.

Therefore, in order to provide a consistent approach for any type of data, aligning with the widely recognized data-agnostic explainer `LIME` (Ribeiro, 2016), we suggest segmenting each input using techniques commonly employed for image inputs. Examples of these techniques include the *Felzenszwalb* approach (Felzenszwalb, 2004), which uses minimum spanning tree-based clustering to segment an image, and *SLIC* (Achanta, 2012), which segments images through k-Means clustering. Additional examples include an extension of *SLIC* called *MaskSLIC* (Irving, 2016), that generates superpixels in specific regions of interest, *Quickshift* (Vedaldi & Soatto, 2008), a local mode-seeking algorithm based on a kernelized mean-shift approximation, as well as *Watershed* (Beucher, 1992), which identifies watershed basins in images flooded from user-given markers.

SINEX does not use a default segmentation algorithm, hereafter referred to as *seg*, as its selection depends on the data type and the context. However, it is flexible as it technically supports all the algorithms described earlier, allowing users to choose based on their specific needs. For a detailed discussion on the suggested procedure to choose the appropriate *seg*, please refer to Appendix C.

Algorithm 1 SINEX(f, x, S)

Input : f - Siamese Network, x - query instance to explain, S - support set
Param : seg - segmentation algorithm, ω - perturbation approach, c - replacing value, γ min nbr. segments
Output: E - explanation

1 $E \leftarrow \emptyset$; // init. explanation
2 **for** $s_i \in S$ **do** // for each support sample
3 $v \leftarrow f(x, s_i)$; // calculate similarity/distance
4 $R \leftarrow seg_\gamma(s_i)$; // apply segmentation
5 $h_i \leftarrow 0^{p \times q}$; // init. sample contribution
6 **for** $r_j \in R$ **do** // for each segment
7 $z_i \leftarrow (s_i[\omega r_j] \leftarrow c)$ // perturb sample
8 $u \leftarrow f(x, z_i)$; // calculate similarity/distance
9 $\delta \leftarrow v - u$; // compute segment similarity/distance delta
10 $h_{ij} \leftarrow \delta / |r_i|$; // weight and update contribution
11 $E \leftarrow E \cup \{h_i\}$; // add heatmap to explanation
12 **return** E;

4.2 SINEX basic approach

In the following, the term "similarity" refers to the outcome of a 2SNs, meaning that the function f to explain is a 2SNs. Conversely, "distance" is used when f is a 3SNs. In both the pseudo-code and the subsequent description, "similarity/distsance" is used to generalize, as the supported f can be either a 2SN or a 3SN. Details on the support for such different architectures will be provided later in this section. The pseudo-code of the basic version of SINEX is reported in Algorithm 1, and detailed as follows. For each sample in the support set S (lines 2–11), SINEX calculates the initial similarity/distance v returned by the application of the SN f between the query input x and the support sample s_i (line 3). Then, it segments

the support sample s_i using the *seg* segmentation algorithm (line 4), resulting in a set of segments R with at least γ segments. For each segment r_j, its contribution is stored in the saliency map h_i, which represents the importance of different areas in the support sample s_i, and it is computed as follows. The saliency maps are initially set to a matrix with all values equal to 0, with the same dimensionality as the support set sample (line 5). The support set sample s_i is then perturbed (line 7) based on the selected perturbation approach ω, which can assume two values. If the $\omega{=}\neg$ perturbation approach is used, everything except the region r_j is obscured. In this case the notation of line 7 becomes $s_i[\neg r_j] \leftarrow c$, that indicates that the support sample s_i assumes value c in all regions except r_j, i.e., we obtain the same Algorithm described in Fedele et al. (2022). Otherwise, if $\omega{=}1$, only the region r_j is obscured. In such case, notation at line 7 becomes $s_i[r_j] \leftarrow c$ indicating that, the support sample s_i assumes the value c only in the r_j region. The value c in Algorithm 1 line 7 symbolizes a replacing value to be used when perturbing, which varies according to the sample data-type. For RGB images, any color might be a replacing value and it should be carefully chosen not to create out-of-distribution samples. In our study we use the gray color, which is commonly used for c in this case. Grayscale images are instead limited to only use color belonging to the grey-scale (from white to black). For this kind of input, we use the white as a replacing color for black pixels. Whenever dealing with audio spectrograms, a reasonable replacing value for c might be -80 as it is the smallest value in the dB scale in many cases. After perturbation, the new similarity/distance score u is calculated by the application of the SN f between the query sample x and the perturbed support set sample z_i (line 8), and the difference δ between the starting similarity/distance score v and the new score u is determined (line 9). Finally, the difference is weighted according to the size of the current segment $|r_j|$ and updated in the corresponding saliency map h_i (line 10).

Algorithm 2 SINEXC(f, x, S)

Input : f - Siamese Network, x - query instance to explain, S - support set
Param : seg - segmentation algorithm, ω - perturbation approach, c - replacing value, α - per-segment coalitions, β - per-coalition active segments, γ min nbr. segments
Output: E - explanation

1 $E \leftarrow \emptyset$; // init. explanation
2 **for** $s_i \in S$ **do** // for each support sample
3 $v \leftarrow f(x, s_i)$; // calculate similarity/distance
4 $R \leftarrow seg_\gamma(s_i)$; // apply segmentation
5 $h_i \leftarrow 0^{p \times q}$; // init. sample contribution
6 **for** $r_j \in R$ **do** // for each segment
7 $\Pi \leftarrow makec(r_j, R, \alpha, \beta)$; // make α coalitions with β (non)active segments
8 $\bar{u} \leftarrow 0$; // sums of segment similarities
9 **for** $\pi_k \in \Pi$ **do** // for each coalition
10 $z_i \leftarrow (s_i[\{\omega r_j\} + \omega\pi_k] \leftarrow c)$ // perturb sample
11 $\bar{u} \leftarrow \bar{u} + f(x, z_i)$; // calculate similarity/distance
12 $\delta \leftarrow v - \bar{u}/|\Pi|$; // compute segment similarity/distance delta
13 $h_{ij} \leftarrow \delta/|r_j|$; // weight and update contribution
14 $E \leftarrow E \cup \{h_i\}$; // add heatmap to explanation
15 **return** E;

4.3 SINEX with coalitions

The basic version of SINEX may suffer from well-known problems associated with perturbation-based methods. First, perturbing instances may result in generating *out-of-distribution*

(OOD) data point, which may not guarantee the validity of the similarity/distance measures. To address this issue, one solution is to retrain the model on a dataset that includes the perturbed data points, but this requires additional time resources. Second, measuring the prediction changes of individual segment perturbations may help understand their contribution to the final outcome, but it may ignore the interaction between the segment and the other parts of the input (*isolated effect*). To tackle these challenges, we take into account the few-shot learning context and the fact that we evaluate the model on new classes, which make the similarity networks robust to OODs, since unseen sample might be considered OOD themselves. However, there is still no guarantee of the network f being robust when comparing a query sample x to a perturbed version of the support set sample s_i, irrespective of whether x and s_i belong to known or never-seen-before classes. Hence, our proposal to mitigate these limitations is to evaluate the contribution of a specific segment to the final outcome by considering its *weighted-average* value. Drawing inspiration by Lundberg (2017), we aim for this value to take into account the interplay between the segment in analysis and the remaining others. However, differently from *SHAP*, our context does not allow us to compute "baseline values" based on the training set. Nevertheless, we can adopt an approach similar to LIME (Ribeiro, 2016), where we perturb not only a single segment but also other randomly selected super-pixels in each step. We aim for our explainer to not only assess the impact of a specific segment on the final outcome, but also consider how its interaction with other parts of the sample influences the final similarity/distance score. To do this, we have introduced two parameters, α and β, which control the number of per-segment coalitions to generate and the number of additional segments that must remain active or disabled in each coalition, depending on which ω perturbation procedure is selected. It is important to emphasize that the selection of additional segments to target is random. Hence, we extend SINEX to include coalitions in SINEXC, as outlined in Algorithm 2 and detailed in the following.

Similar to the basic SINEX version, SINEXC calculates the initial similarity/distance v by applying the SN f between the query input x and the support sample s_i (line 3) for each sample in the support set S (lines 2–14). Then, the support sample s_i gets segmented using the *seg* segmentation algorithm (line 4), resulting in a set of segments R with at least γ segments. The contribution of each segment r_j is stored in the saliency map h_i, initially set to a matrix with all values equal to 0, matching the dimensionality of the support set sample (line 5). Unlike SINEX, SINEXC now generates additional α coalitions with β ω-active segments for each r_j segment (line 7). Indeed, each coalition specifies β random segments different from r_j that will either remain active (ω=¬) or non-active (ω=1) along with r_j during the following perturbation procedure. For each coalition π_k (lines 9–11, Algorithm 2), we perturb the support sample s_i based on the selected perturbation approach (line 10). If ω=¬ perturbation approach is selected, line 10 of Algorithm 2 becomes $z_i \leftarrow (s_i[\{\neg r_j\} + \neg\pi_k] \leftarrow c)$, indicating that everything except r_j and the additional set of segments indicated by π_k is obscured. Otherwise, if ω=1, line 10 becomes $z_i \leftarrow (s_i[\{r_j\} + \pi_k] \leftarrow c)$, indicating that only r_j and the additional set of segments indicated by π_k are obscured. Finally, the sum of the similarity scores obtained querying the model is stored in $\bar{u}$. We then compute the segment contribution δ (line 12, Algorithm 2), as the difference between the original similarity score, v, and the mean similarity value $\bar{u}/|\Pi|$ obtained from the $|\Pi|$ coalitions. Lastly, we weight the *segment-average* prediction value according to the size of the segment in analysis (line 13, Algorithm 2). This process is iterated pairing the query sample x with every sample of the support set S (line 2–14, Algorithm 2), keeping x fixed and applying the coalition-based methodology to each s_i element of the support set S.

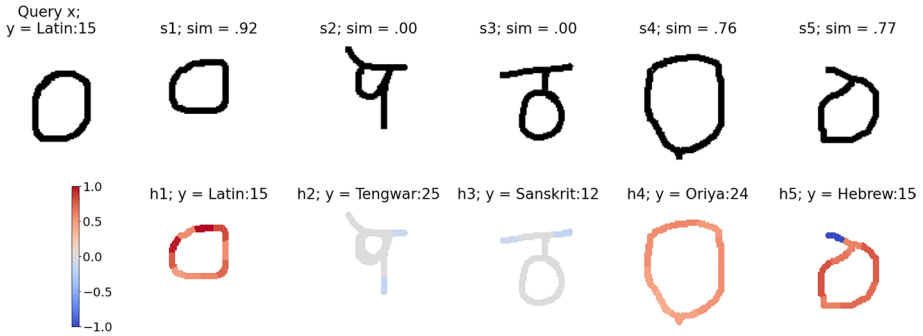

Fig. 4 SINEX explanation on a 5-way 1-shot classification task on a hand-drawn characters dataset. Top row: query sample x followed by the support set samples s_1 to s_5 from left to right. Bottom row: explanation heatmaps h_1 to h_5 corresponding to the s_1 to s_5 support samples. Samples belonging to the *Latin* (s_1), *Oriya* (s_4), and *Hebrew* (s_5) classes exhibit visual similarities. A segment in the top left corner of s_5 is identified with a negative impact, as s_5 would be more similar to the query x in its absence. Best viewed in colors (Color figure online)

4.4 SINEX explanation

Figure 4 illustrates an example of a SINEX/SINEXC explanation E that contains the saliency maps $\{h_1, \ldots, h_C\}$ (bottom row in Fig. 4) for each support set sample $\{s_1, \ldots, s_C\}$ (top row in Fig. 4). Each h_i holds the contribution value for each segment of the support sample s_i, whether it is a positive or a negative value. These contribution values are normalized for comparability across all $h_i \in E$ within each *C-way one-shot* task. In case of similarity based approaches, the scale is normalized in $[-N, +P]$, where N is the maximum value within the negative contributions and P is the maximum value of positive contributions instead. Vice-versa, the min-max scale for distance-based approaches is in $[-P, +N]$. This normalization process is intentionally tailored for each specific *C-way one-shot* task, enabling meaningful heatmap comparisons within the confines of that task. Thus, it only allow relative comparisons across different explanations, since each one is generated on distinct query sample x and support set S samples.

A color-map must be selected to visualize E such that the contribution values close to 0 are non influential to the final prediction. We adopt a *blue-to-red* color-map where negative contributing segments range from dark blue (stronger influence) to light blue (lower influence) and positive contributions range from light red (lower influence) to dark red (stronger influence). Non influential pixels are colored in gray. To respect the definitions of positive and negative colors, an inverse color-map is applied for the distance-based approach.

Combining such dual min-max scale for normalization and the inverse color-map for visualization, we ensure that the cognitive workload of the end-user watching the explanation's heatmap is oriented to a unique semantic meaning of positive or negative influencing segments. Thus, regardless of the approach being based on similarity or distances, SINEX outputs the same visual result.

4.5 SINEX support for 2SN and 3SN

The SN architecture comprises two primary components. The first one transforms the inputs into an embedding space, while the second component evaluates the proximity of

the embedded inputs. The first component can either be implemented as a 2-input-branch (2SN) (Koch et al., 2015) or a 3-input-branch (3SN) (Hoffer, 2015) network, with the branches responsible for input embedding. Once the inputs are embedded, the second component assesses their proximity, producing either a similarity or distance score for 2SNs and 3SNs, respectively. In our study, we use both 2SNs and 3SNs. 2SNs are explored by leveraging their similarity scoring output, which is more commonly described in the literature. We also consider 3SNs due to their superior discriminative performance compared to 2SNs, especially in challenging tasks. While 2SNs produce similarity scores, 3SNs generate distance values. Throughout the manuscript, similarity will refer to an underlying architecture in the form of 2SNs, while distance will be used for 3SNs. In 2SNs, the query input x and the support set instance s_i are provided to the network, and the changes in the similarity score are captured after each s_i perturbation run. On the other hand, 3SNs take three inputs and are typically trained using a Triplet Loss function (Schroff et al., 2015), which enforces that dissimilar pairs are separated by a certain margin compared to similar pairs. The three inputs for the network are the query instance x, a *positive* sample $x_=$ from the same class as the query, and a negative sample $x_{\neq}$. 3SNs are designed to measure the distance score between $< x, x_= >$ and $< x, x_{\neq} >$ pairs independently, by means of the same distance function and prior to the application of the triplet loss function. This creates a point of attachment for our explainers within the 3SNs architecture, which is not necessary in 2SN. In order to use SINEX within this context, $x_{\neq}$ is disregarded and the support sample s_i is represented by the positive sample $x_=$ in the $< x, x_= >$ pair distance scoring.

5 Experiments

This section describes the experiments we conducted on five different datasets to validate SINEX and SINEXC, both implemented in Python.[2] After presenting the experimental setting we report a qualitative and quantitative evaluation of the explanations. Then, we analyze SINEX and SINEXC using both the ω=¬ and ω=1 perturbation procedures. We also compare SINEX against conventional explainability techniques to assess their capability in identifying informative segments within the broader SN architecture. Finally, we further investigate positive and negative contribution segments using the novel WPE and WNE metrics.

5.1 Experimental setting

We selected five diverse dataset for classification, encompassing two image datasets, namely *Omniglot* (OGT) and *Caltech-UCSD Birds 200* (CUB), a time-series dataset known as *FiftyWords* (50W), and two audio datasets, *AudioMNIST* (AST) and *ESC-50* (ESC). Our selection of these datasets was guided by the aim of developing few-shot learning models, a task that is greatly enhanced by the presence of a large number of classes during training. In Table 1 we report a comprehensive overview of the datasets showing the total number of records, their data-type, and their dimensions before and after the pre-processing. For consistency across all datasets, we utilized five classes for validation and another five distinct classes for testing. Importantly, the testing classes were entirely separate from the training

[2] https://github.com/andreafedele/SINEX.

Table 1 Left: characteristics of the datasets and their dimensions before (*pre-dims*) and after our processing (*post-dims*), with n being the total number of records, and L the number of classes used at training time

Dataset	Type	Pre-dims	Post-dims	n	L	c_1	c_2	c_3	c_4	c_5	Avg. acc
OGT	Gs-image	105×105	$105 \times 105 \times 1$	32k	40	.91	.88	.94	.97	.95	.93
CUB	Rgb-image	500×500	$224 \times 224 \times 3$	3,6k	50	.85	.80	.70	.70	.55	.72
50W	Time series	270 points	$114 \times 114 \times 1$	1k	40	.73	.91	.91	.99	.76	.86
AST	Audio	1 sec	$224 \times 224 \times 1$	30k	50	.93	.91	.88	.82	.78	.86
ESC	Audio	5 sec	$128 \times 431 \times 1$	2k	40	.99	.93	.91	.82	.71	.87

Right: individual accuracy for the 5-way 1-shot classification for each class, and the average accuracy

classes, ensuring a robust evaluation of our models. Details about the datasets and the pre-processing we performed are available in Appendix A.

We tailored the choice of SN architecture to align with the characteristics of each dataset. For AST, ESC, OGT and 50W datasets, following established practices (Koch et al., 2015; Honka, 2019; Acconcjaioco, 2020; Zhang, 2019), we employed a 2-branched SN with slight variations to suit each dataset's unique features. These 2SNs consist of two convolution-based encoders, a distance layer, and a final output similarity scoring layer. In contrast, for the CUB dataset, due to the more challenging discriminatory nature (3 RGB channels), we implemented a 3SN architecture, in line with (Hoffer, 2015). This 3SN architecture includes three encoding-branches sharing the same architecture and weights between them. It differs from the 2SNs in terms of output and training requirements. Specifically, the 2SNs used binary *Binary cross-entropy* loss for classifying input pairs as belonging to the same class (label 1) or different classes (label 0). On the other hand, the 3SN used *Triplet Loss* (Schroff et al., 2015) with a margin of 0.5 to encourage a clear separation between similar and dissimilar input pairs. More details regarding the architectures and training processes can be found in Appendix B.

We evaluated the SN performance by conducting assessments every 100 training epochs on 300 randomly generated 5-way 1-shot tasks. In each task, the model's objective was to classify a given query input, denoted as x, by comparing it with each support sample, $s_i \in S$ (as described in Sect. 3). Successful classification occurred when the class, y_i, of the support sample s_i with the highest similarity score (or lowest distance) matched the class of the query input x. Importantly, all evaluations were performed using sets that exclusively contained unseen classes, ensuring that no samples from the test classes had been encountered during the model's training. We terminated the training procedure for all datasets when the model did not exhibit an improvement in 5-way 1-shot accuracy for 10 consecutive evaluation runs. Table 1 shows the mean *5-way one-shot accuracy* for each dataset, as well as the accuracy on each singular class, to provide a complete overview of the model's performance. Although the goal of this study is not to develop the best-performing SN in the given settings, we relied on average accuracy which can be considered satisfactory in 5-way 1-shot learning. The final mean accuracy of the SNs used in this study is in line, if not better (ESC (Honka, 2019)), than state-of-the-art networks in the same setting.

To maintain consistency across experiments, we used a uniform setting for the replacing value (c parameter in Algorithm 1 and Algorithm 2). This involved selecting a silence value of $-80dB$ for spectrograms (AST and ESC), a white background for grayscale images and coverted time series (OGT and 50W), and a gray RGB color [128, 128, 128] for CUB. The segmentation algorithms, referred to as the *seg* in Algorithm 1 and Algorithm 2, and

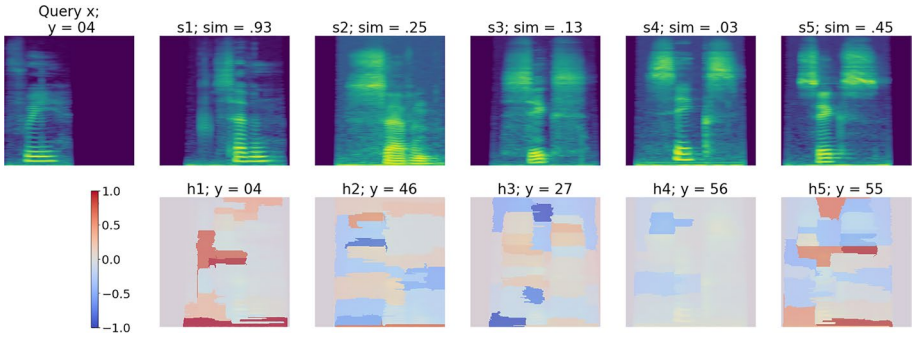

Fig. 5 SINEX$_1$ explanation on a 5-way 1-shot task on the AST dataset. Class labels of the test set are composed of one female speaker ($y = 56$) and four male speakers. Best viewed in colors (Color figure online)

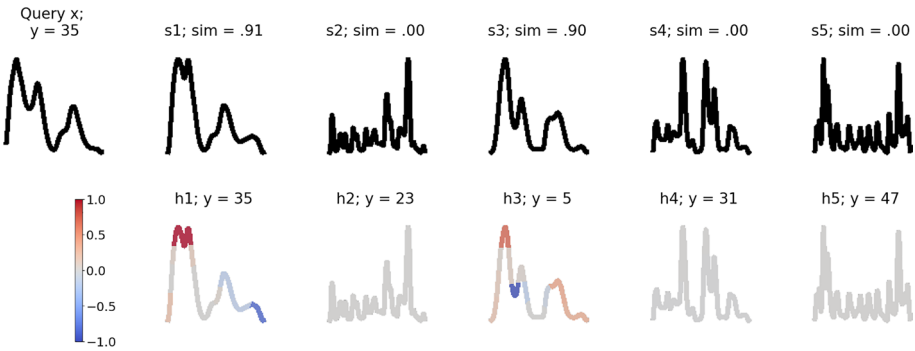

Fig. 6 SINEX$_1$ explanation on a 5-way 1-shot task on the 50W dataset. Each time-series represents the height profile of a written word. Best viewed in colors (Color figure online)

their parameter configurations were selected according to the specific data type. Appendix C provides a detailed description of the procedure that guided us in this selection and the parameters chosen for all experiments. Lastly, we set the SINEXC α parameter at 200 for all experiments. Preliminary tests indicated that increasing this value further did not yield any improvement in terms of measured performance and would only negatively impact run times. Also, in Appendix G, we show how SINEX explanations remain coherent across different sample augmentations, provided that the SN's performance is not affected by the shifting.

5.2 Qualitative evaluation

Figures 1, 5, and 6 showcase examples of SINEX explanations on CUB, AST, and 50W, respectively.[3] We recall the reader that blue areas represent segments of negative influence, while red portions indicate segments that positively affect the similarity score outcome. Grey areas are instead neutral to the SN classification process.

[3] Explanation obtained from SINEX$_1$ on CUB and AST, and SINEXC$_1$ on 50W. Additional parameters are listed in Appendix C.

Concerning Fig. 1, SINEX highlights that the correct classification of the *Rose breasted Grosbeak* class in CUB depends primarily on the bird's red breast color, while there is a possible miss-classification error towards the *Summer Tanager* class due to their similar red body color. This error is more likely to occur when the support set sample of the *Rose breasted Grosbeak* class contains a distant bird, making it difficult for the network to identify the red breast, and leading it to rely more on the full red body of the *Summer Tanager* bird. An example of such miss-classification is illustrated in Fig. 11 in Appendix E.

Figure 5 shows an example of explanation for AST. The query class represents a male produced audio scoring a .93 similarity value for the correct s_1 support sample, despite triggering both s_2 and s_5 so to reach .25 and .45 scores respectively. The explanation helps in understanding that s_1 is mainly composed of positive segments, which have an absolute influence bigger than the positive segments of the other two samples. In addition, samples s_2 and s_5, present spectrogram portions which get marked as negatively impacting on the model outcome, therefore decreasing the similarity score value. Using SINEX, we analyzed several 5-way one-shot tasks for AST and found that correct classifications of male/female speaker recordings depend mainly on medium-high/low frequency segments respectively, while miss-classifications are primarily due to segments at the opposite end of the frequency spectrum. An example of such miss-classification is illustrated in Fig. 13 in Appendix E. Additionally, unlike our previous study where $\omega{=}\neg$ perturbation procedure is discussed (Fedele et al., 2022), we found that setting $\omega{=}1$ removes the issue of explanations relying on silent areas in spectrograms.

In Fig. 6, we report an explanation of SINEX for 50W, which reveals why the SN correctly classifies the query class labeled as *35* and highlights possible miss-classifications with class *5*. The support sample s_1, which belong to the same class of the query sample x, achieves a .91 similarity score, thanks to a distinctive up-down-up trend at the beginning of the time-series. In contrast, s_3 has a more relaxed curve drop that resembles the query input, which could lead to miss-classification with class *5*. This is a lucky case example for s_1, since s_3 reaches a very close similarity score of .90. Since $y = 35$ and $y = 5$ classes are very similar, the form of the query sample is fundamental for the final classification. We found that most of the $y = 35$ time series share the fast up-down-up trend of s_1, but the variability of word outlines can make them more similar to other classes. An example of such miss-classification is illustrated in Fig. 12 in Appendix E.

5.3 Quantitative evaluation

We evaluated the qualitative significance of the explanations generated by SINEX following the methodology described in Petsiuk (2018). In particular, we calculated the insertion and deletion scores by incrementally adding or removing the most influential pixels identified by our explainers, starting from an empty or full object, respectively. We expect the insertion curve to exhibit a rapid increase after inserting only a small percentage of relevant pixels, resulting in a large insertion-area-under-curve *iAUC*. Conversely, we expected the deletion curve to exhibit a rapid decrease after only a few pixel removals, resulting in a small deletion-area-under-curve *dAUC*. A value close to one for *iAUC*, or close to zero for *dAUC* indicates that the explainer successfully identifies the important pixels for the classification process (Petsiuk, 2018). As a preliminary experiment, we studied the impact of β on SINEXC. Detailed results are available in Appendix D. Based on these findings, we identified the β prioritizing a balance between high *iAUC* and low *dAUC* for each dataset

and used them in the following experiments. Our selections, is listed in Table 4 in the same Appendix. In our evaluation, we included the Gradient-weighted Class Activation (GRAD-CAM) and Epsilon Layer-wise Relevance Propagation (ϵ-LRP) techniques (Selvaraju, 2020; Bach et al., 2015) for comparison. Differently from SINEX, GRAD-CAM utilizes a gradient-based approach, and it is commonly employed to analyze how convolutional based neural networks break down matrix-like inputs (Moujahid et al., 2022; Majid et al., 2022) at different convolutional stages. In our study, we use GRAD-CAM on the last layer of the CNN responsible for embedding, as it captures the final attention towards the input. The LRP technique explains predictions by back-propagating the outcome through the model to assign a relevance score for each layer, employing specifically designed propagation rules. In our application of the LRP technique, we utilize the ϵ rule, referred to as ϵ-LRP henceforth, with $\epsilon = 1$, targeting the entire CNN for enriched explainability. While GRAD-CAM and ϵ-LRP are established techniques in the literature, they do not provide complete explanations for the SN as SINEX does, as they cannot be applied on the distance layer. Therefore, these methods should not be considered "proper baselines" in the conventional sense, but rather comparisons for XAI techniques in the context of few-shot learning. The purpose of this comparison is to assess the degree of alignment between the features emphasized by the embedding networks and those utilized by the SN in its final scoring process. We compared the performance of SINEX with its coalition version, SINEXC, using both ω=¬ and ω=1 perturbation procedures. The rationale for this comparison lies in assessing how different perturbation methods affect the explanation generation process. We measured the mean *iAUC* and *dAUC* values for 150 5-way one-shot tasks per dataset, split it into 30 experiments for each of the 5 test classes. The results in Table 2 compare $\text{SINEX}_{\neg}$, SINEX_{1}, $\text{SINEXC}_{\neg}$, SINEXC_{1}, GRAD-CAM and ϵ-LRP for the same *one-shot* tasks. Bold highlights the best scores, aiming for high *iAUC* and low *dAUC*. Also, Fig. 7 presents the average *iAUC* and *dAUC* curves for each dataset, showcasing the performance of the different explainability algorithm tested.

Our analysis points to ω=1 as the more effective perturbation procedure. Specifically, SINEX_{1} outperforms other variations, showing excellent performance on four out of five datasets. While its coalition counterpart, SINEXC_{1}, surpasses it marginally only in 50W, the difference is not significant. For the most part, ω=1 consistently yields satisfactory *iAUC* and *dAUC* results. Notably, it excels in CUB and AST, with commendable performance in ESC and 50W. Despite having the best *dAUC* among all datasets, OGT exhibits the poorest performance in terms of *iAUC*, with the lowest value compared to other datasets. This could be attributed to the distinctive nature of OGT, featuring black characters on a white background. The immediate increase in *iAUC* is not attained, as, at initial insertion percentages, the majority of the sample comprises only the background color. Consequently, a substantial number of black pixel insertions is required to convey valuable information to the model. This behavior is somehow reflected in 50W, which consists of black lines on a white background. Despite having the best *dAUC* among all datasets (0.14), the *iAUC* value for 50W is the second lowest after OGT. In general, SINEXC_{1} does not show any significant improvement over SINEX_{1}. However, the good performance of SINEXC_{1} on 50W suggests that coalition-based perturbation techniques could be useful in certain scenarios.

Turning our attention to ω=¬, we observe a decline in performance compared to the ω=1 configuration, affecting both SINEX and SINEXC. The insertion and deletion curves lack the desired abrupt rise and fall, suggesting a less than ideal perturbation approach. This observation is consistent with the results of our quantitative β analysis in Appendix D. Similarly to the findings under ω=1, we do not observe significant enhancements in *iAUC* or *dAUC* when SINEXC is used in the ω=¬ setting. This suggests that to discover a given

Table 2 Mean *iAUC*, *dAUC*, and execution time (in min) for the same randomly generated 150 tasks of 5-way one-shot learning for each of the five listed datasets

	SINEX$_{\neg}$			SINEX$_1$			SINEXC$_{\neg}$			SINEXC$_1$			GRAD-CAM			ϵ-LRP		
	iAUC	*dAUC*	*Time*	*iAUC*	*dAUC*	*Time*	*iAUC*	*dAUC*	*Time*	*iAUC*	*dAUC*	*Time*	*iAUC*	*dAUC*	*Time*	*iAUC*	*dAUC*	*Time*
OGT	.28	.32	0.01	**.42**	**.14**	0.01	.28	.32	0.15	.41	.15	0.15	.13	.17	2e−9	.49	.52	0.03
CUB	.44	.86	0.08	**.95**	**.19**	0.08	.44	.86	1.30	.85	.24	1.30	.89	.55	1e−8	.78	.60	0.06
50W	.35	.44	0.03	.70	.18	0.03	.30	.45	0.50	**.70**	**.14**	0.50	.66	.16	5e−9	.35	.63	0.04
AST	.42	.57	0.05	**.85**	**.19**	0.05	.47	.44	0.60	.76	.19	0.60	.62	.74	8e−9	.85	.40	0.06
ESC	.69	.66	0.04	**.88**	**.39**	0.04	.70	.56	0.50	.88	.40	0.50	.73	.60	7e−9	.57	.32	0.06

The best results are indicated in bold when both high *iAUC* and low *dAUC* are achieved simultaneously

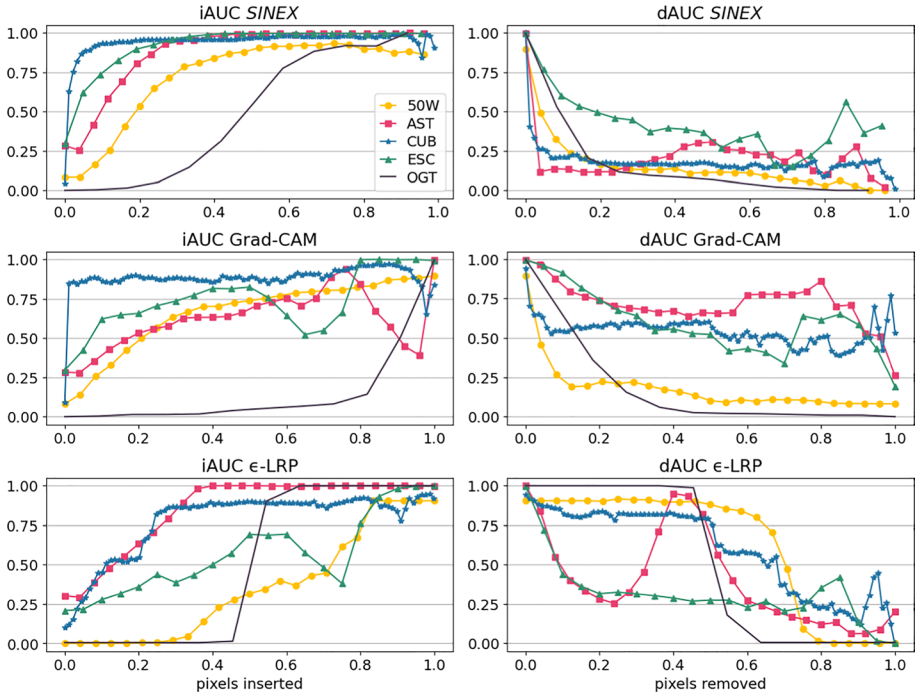

Fig. 7 Insertions (left) and deletions (right) curves for SINEX_1 on AST, ESC, 50, CUB and SINEXC_1 on 50W in the first row. Second and third rows respectively display curves for GRAD-CAM and ϵ-LRP on the same datasets (Color figure online)

input-segments contribution to the final outcome, it is much more effective to deactivate the segment while maintaining intact the context in which it is included (ω=1), rather than deactivating the whole context to only keep the segment in analysis as active (ω=¬). This behavior, which aligns with human intuition when evaluating the similarity between objects, is mirrored in the ω=1 perturbation approach of our explainer.

Furthermore, when comparing to the ω=1 configuration, GRAD-CAM and ϵ-LRP typically demonstrate poorer performance across various datasets. Exceptions include 50W, where GRAD-CAM performs well in both measures simultaneously, and in the cases of OGT-*dAUC* and CUB-*iAUC* separately. These results indicate that the pixel areas considered interesting for the embedding sub-network within the SN architecture, according to GRAD-CAM and ϵ -LRP, may not be essential for the SN's overall scoring. This conclusion is in line with our understanding that SINEX is designed to provide a comprehensive explanation of the entire SN, while GRAD-CAM and ϵ-LRP are not.

The runtime performance analysis of the compared explainability methods reveals that SINEX outperforms its coalition version, being 12 times faster across all datasets. This observation is intuitive, given that SINEXC has to navigate through the extra α coalition perturbations. Notably, the perturbation procedure ω does not impact the execution time for either SINEX or SINEXC. While the GRAD-CAM technique stands out as the fastest overall, it comes at the expense of the measured performances in terms of *iAUC* and *dAUC*. This efficiency in GRAD-CAM's runtime is primarily attributed to its gradient-based approach, requiring only forward and backward passes through the CNN. On the other hand, SINEX and SINEXC are

perturbation-based approaches, measuring the output difference after a set of perturbations. Such perturbation-based approaches are generally slower than gradient-based ones, as confirmed by our experiments. The slowest execution times for both SINEX and SINEXC are observed for the largest dataset CUB (3 channels). Additionally, the increased execution times on the CUB dataset can be attributed to its use of a 3SN, which is computationally more intensive than a 2-branched SN. This is due to the presence of three duplicated embedding CNNs and three inputs, as opposed to two in the 2SN. Interestingly, SINEX exhibits similar time performance to the gradient-based technique ϵ-LRP. However, we remark that both SINEX and SINEXC have potential for time-performance improvement. Indeed, in SINEX, the perturbations of individual segments (Algorithm1, line 7) are independent and could be parallelized. Similarly, in SINEXC, perturbations of single segments (Algorithm2, line 10) could be computed in parallel once α coalitions are generated and shared (Algorithm 2, line 7). Also, it is important to note that the time-performance is heavily reliant on the segmentation algorithm *seg*. Therefore, a preliminary assessments to select an appropriate segmentation algorithm and parameter settings is recommended. In Appendix C we discuss our proposal for selecting the segmentation technique *seg* for different data types.

Table 2 shows that the optimal configurations for each dataset are SINEX_1 for AST, ESC, OGT, and CUB, and and SINEXC_1 for 50W. When examining the *iAUC* curves of SINEX or SINEXC, we observe a consistent and smoother trend across all datasets, differing from both GRAD-CAM and ϵ-LRP behavior. The SINEX or SINEXC *iAUC* curve display a swift rise in similarity scores, followed by stabilization. Conversely, GRAD-CAM exhibits slower responses and is more prone to perturbations, particularly when a high percentage of pixels is inserted. This behavior may arise because the pixels highlighted by GRAD-CAM as crucial for the embedding process might not accurately represent those essential for SN similarity scoring. For instance, consider the AST *iAUC* curve in GRAD-CAM. It experiences a decline as the last 20% to 10% of remaining pixels are inserted. However, when the last 10% of pixels, i.e., those considered less important by GRAD-CAM, are inserted, the SN's similarity score rebounds from 0.4 to 1. This suggests that the pixels GRAD-CAM deems of marginal importance may exert a mixed influence, being very negative from the last 20% to the last 10% and very positive from the last 10% to the complete picture. Insertion curves for the ϵ-LRP technique, generally show a slower rise than GRAD-CAM ones, but are less prone to perturbation at high percentage of pixels insertion. Indeed, in the insertion curves of ϵ-LRP, there is no dataset where significant drops and subsequent rises occur after inserting more than 80% of pixels, as observed in GRAD-CAM. However, such slow rise of insertion curves is the slowest in ϵ -LRP if compared to both GRAD-CAM and SINEX and it is mainly due to the fact that ϵ-LRP highlight as the most positive influential pixels areas that are typically at the border of the sample semantic value (i.e, the drawn line in OGT and 50W or the audible portion of high dB value in audio data). For instance, when applied to OGT, ϵ-LRP highlights as important the background pixels that are at the boundaries of the written character, rather than the character itself. The entire character is considered neutral, leading to an immediate peak where approximately 0.5% of the pixels are inserted. Remarkably, even for the worst-performing *iAUC* curve for the OGT dataset, SINEX still outperforms both GRAD-CAM and ϵ-LRP.

In terms of *dAUC*, SINEX exhibits smoother trends compared to GRAD-CAM and ϵ-LRP. All *dAUC* curves, except for ESC, show a rapid drop even before 0.2% of pixels are removed, indicating very good performance. On the other hand, GRAD-CAM only performs well for 50W and OGT datasets, with poor results for the remaining three datasets, which exhibit a much slower drop and are highly susceptible to perturbations at high percentages of pixels removed. Deletion curves for ϵ-LRP are not satisfactory either, since they all seem to achieve the desired drop only after that 50% of the pixels are removed. The best ϵ-LRP

deletion curve is on the ESC dataset, which drops earlier than SINEX and stabilizes at lower similarity scores. In this case, relevant pixels align with the audible portions of the sample, characterized by higher dB values, while the background has minimal influence. Unfortunately, ϵ-LRP explanations for the ESC dataset are quite sparse with both positive and negative pixels scattered across the entire spectrum, therefore not conveying clear semantic insights. The superior performance of ϵ-LRP on the ESC dataset compared to SINEX may be attributed to the fact that SINEX relies on super-pixel perturbations, potentially overlooking smaller, equally important pixels scattered sparsely across the sample. The least favorable deletion curve among all tested explainability methods is observed with ϵ-LRP on the AST dataset. The curve displays two valleys and one peak when removing 20% to 40% of pixels. This suggests that the order of importance returned by ϵ-LRP in this case may not align with the actual pixel importance within the samples.

In general, ϵ-LRP performs worse in terms of both *iAUC* and *dAUC* compared to GRAD-CAM. Furthermore, while GRAD-CAM achieves acceptable results only for 50W, ϵ-LRP consistently fails to exhibit high *iAUC* and low *dAUC* simultaneously. Our quantitative analysis indicates that SINEX performs exceptionally well when combined with the ω=1 perturbation approach. This not only enhances its explanatory capabilities but also makes it a suitable choice for real-time applications requiring swift model explanations.

5.4 Dependence between positive and negative segments

We delved into an exploration of the dependence between positive and negative contributing segments within SINEX. The motivation behind this choice arises from the observation that SINEX demonstrates smoother *iAUC* and *dAUC* curves than ϵ-LRP, implying a more precise assessment of the importance order of various segments. Unlike SINEX and ϵ-LRP, GRAD-CAM does not identify positive and negative contributing pixels and is therefore excluded from this analysis. Specifically, we aimed to discern whether positive segments derive their positive influence solely from the presence of their negative counterparts, or if the reverse holds true. To investigate this, we introduced two novel metrics, WPE and WNE, building upon the insertion and deletion processes outlined in Petsiuk (2018). WPE, short for *Whole input* $\longleftrightarrow$ *Positive only segments* $\longleftrightarrow$ *Empty input* , involves a two-step deletion process and a two-step insertion process. The deletion process commences with the entire input and progressively removes negative segments to obtain an intermediary state containing only positive segments. Subsequently, all positive segments are removed to arrive at an empty sample. In contrast, the insertion process starts with an empty input and introduces positive segments followed by negative ones to restore the original input. Conversely, WNE stands for *Whole input* $\longleftrightarrow$ *Negative only segments* $\longleftrightarrow$ *Empty input* . In this scenario, the intermediary state requires solely negative segments, and the deletion process begins with the whole input, removing positive segments first and subsequently eliminating the remaining negative segments. The insertion process starts with an empty sample and introduces negative segments followed by positive segments to reconstruct the original input. In our insertion and deletion framework, we always add or remove segments from the most to the least influential at each intermediary step, regardless of whether we are using WPE or WNE. The idea is that by analyzing the area under the curve at each intermediary step, we can better determine the dependence between positive and negative segments. The areas under the curve are referred to as $dAUC\underline{X}_w$ and $iAUC\underline{X}_w$ indicating the deletion/insertion of negative segments for $\underline{X} = N$, or positive segments for $\underline{X} = P$, and $w \in \{\text{WPE}, \text{WNE}\}$. By

Table 3 Deletion and insertion area-under-the-curve (AUC) for WPE and WNE for CUB, 50W and AST on both correct and incorrect 5-way one-shot classification tasks

		WPE			WNE				
	Classification	dAUCN ↑	dAUCP ↓	iAUCP ↑	iAUCN ↑	dAUCP ↓	dAUCN ↓	iAUCN ↓	iAUCP ↑
CUB	*Correct*	.988	.169	.958	.898	.078	.197	.104	.906
	Incorrect	.967	.245	.892	.847	.106	.161	.155	.906
50W	*Correct*	.944	.128	.729	.842	.140	.018	.052	.645
	Incorrect	.829	.099	.625	.585	.120	.015	.072	.639
AST	*Correct*	.965	.221	.736	.871	.083	.151	.107	.627
	Incorrect	.930	.296	.714	.696	.081	.133	.087	.625

↑, ↓ indicate if a measure is preferred to be high or low, respectively

examining these areas, we can gain insight into the individual impact of positive and negative samples during both the deletion and insertion processes.

Building on the best-performing SINEX/SINEXC settings from our prior analysis, we conducted 300 experiments for each dataset, including CUB, 50W, and AST. These experiments were categorized into thirty correct and thirty incorrect 5-way one-shot tasks for each of the five test classes. We remind the reader that a correct classification occurs when the class y_i of the support sample s_i, which achieves the highest/lowest similarity/distance score, matches the class of the query input x. Otherwise, an incorrect classification occurs. These experiments differ from previous ones for two main reasons. Firstly, we measure the resulting area-under-the-curve separately for positive and negative contributing segments, following a specified order (Whole input $\longleftrightarrow$ Positive only segments $\longleftrightarrow$ Empty input for WPE and Whole input $\longleftrightarrow$ Negative only segments $\longleftrightarrow$ Empty input for WNE). In contrast, previous experiments considered the order of positive segments first and then negative segments, regardless of insertion or deletion procedures. Notably, the insertion procedure for WPE and the deletion procedure for WNE remain consistent with the previous experiments. Secondly, these experiments include tasks of miss-classification, whereas previous experiments only focused on correct classifications. Our results, presented in Table 3, showcase mean area-under-curve values, demonstrating consistent outcomes across datasets and between correct and incorrect classification tasks within the same dataset. This consistency implies the robustness of the segments identified by SINEX, capable of pinpointing both true positive and true negative influencing segments, irrespective of the classification accuracy. In Fig. 8, we present the mean WPE and WNE insertion and deletion curves for CUB, reflecting the results from 300 5-way one-shot tasks. Comparable curves for 50W and AST are provided in Appendix F.

Our observations for WPE reveal high *dAUCNwpe* and low *dAUCPwpe* values, signifying that removing only negative segments initially does not lead to a curve drop and instead elevates the predicted similarity scores, aligning with the definition of *negative segments* in Sect. 4. Subsequent removal of positive segments causes the desired curve drop, reflected in low *dAUCPwpe*. On the contrary, the addition of important positive segments results in high *iAUCPwpe* and an immediate curve increase. The curve typically stabilizes or experiences a minor increase after the introduction of negative segments, as indicated by $iAUCN_{wpe}$. In contrast, the results for WNE exhibit an immediate drop in low *dAUCPwne* as soon as positive segments are removed. The curve may stabilize, increase, or decrease when negative segments are removed, as shown by *dAUCNwne*, implying that negative

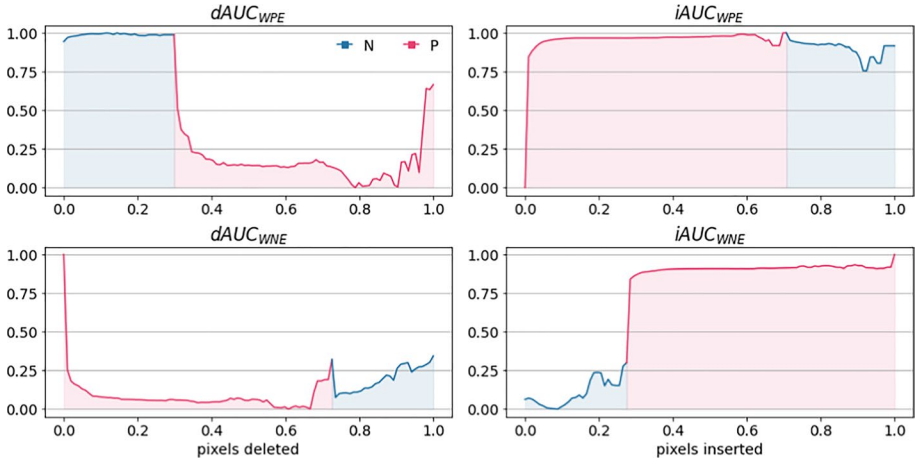

Fig. 8 Deletion (left) and insertion (right) curves for WPE (top row) and WNE (bottom row) procedures on the CUB dataset. Blue colored portions indicate the removal or insertion of negative segments. Red colored areas represent positive segments. Best viewed in color (Color figure online)

segments do not exhibit a clearly defined behavior or a strong influence when they are the sole segments present in the sample. The behavior is also evident in *dAUCNwne* curves for 50W and AST, presented in Figs. 14 and 15 in Appendix F. Low *iAUCNwne* further supports this aspect, demonstrating an immediate curve rise as soon as positive segments are inserted, as indicated by high $iAUCP_{wne}$.

In summary, these findings suggest that negative segments rely on positive ones, while positive segments operate independently of the negative ones. Combining SINEX and the dependence analysis, developers can be better guided in refining the training of SNs. If positive segments are independent and have the greatest influence on the classification, and a miss-classification is caused by positive segments, there might be the need of fine-tuning the corresponding features that these segments focus on, i.e., the red color of the bird's breast. However, retraining on areas indicated by negative segments may not be necessary since they do not have a strong impact.

6 Discussion

The qualitative evaluation of the explanations demonstrates that SINEX is effective in uncovering limitations that SNs might encounter. For example, it can reveal the erroneous dependence of the model on specific colors in the case of RGB images or pixels for grayscale images that should not be considered as important features. For instance, we can consider the model's reliance on the red color in the in the CUB dataset presented examples. To address this issue, a potential approach might consist in training the SN using color masks selected through SINEX on the training samples to avoid such biases. Additionally, augmentation techniques, such as rotating the training samples, can be effective in preventing the development of strong associations between a class and a specific image region, like the bird's breast, which may appear in different locations across various samples. These suggestions, which stem from a post-hoc SINEX-guided analysis, can be addressed in two ways. The first approach involves retraining the SN from scratch with an

augmented version of the original training set. The second approach is a subsequent fine-tuning phase using an additional dataset that targets the limitations highlighted by SINEX, such as augmented samples and color masks. In both cases, SINEX can guide in deciding the augmentation technique to use.

The quantitative evaluation using *iAUC* and *dAUC* scores is an effective method for assessing the explainer's performance. This approach might reveal, for instance, potential weaknesses that SNs may encounter when working with grayscale images. The `OGT` and `50W` datasets exhibited the lowest *iAUC* values, scoring 0.42 and 0.70, respectively, while both achieved the highest *dAUC* score of 0.14. These results suggest that the explainer can indeed identify important pixels, as removing them individually leads to a decrease in deletion scores. However, introducing only a small portion of important pixels onto a white background still hinders the SN from achieving the expected high similarity score. This behavior may indicate that grayscale images pose a more challenging task for SNs, or additional training may be necessary. For instance, in the `OGT` dataset, training samples could be augmented by generating unfilled character outlines. This augmentation phase would enable the network to learn character shapes while emphasizing the concept of a prominent white background. By doing so, the SN can not only capture the complete character shape but also consider that the background has a significance.

While SINEX serves as a valuable tool for comprehending why SNs classify a query sample with specific labels, its explanations remain localized, addressing each *C-way k-shot* independently. Therefore, human examination is necessary to integrate SINEX explanations and gain a broader understanding of the network's behavior. With respect to its current stage, it is left to human-agents to go through different explanations to gain a more global insight of the network's behavior. Explainability is crucial for establishing trust in few-shot learning models, ensuring their safe deployment in real-world applications. For example, we can consider the context of a few-shot learning intrusion detection system for railway video surveillance, as examined in Gong et al. (2021). While the authors did not develop it using SNs, such an approach remains entirely feasible. In such a scenario, SINEX can help evaluate the system's robustness. For instance, consider a situation where miss-classifications of night intruders are associated with shadows in the pixel area related to the sky. Through a SINEX analysis of the system's classifications, segments of pixels related to these shadows might be revealed. Upon human evaluation of SINEX explanations, a decision may be made that it is not safe to deploy the system in its current state.

7 Conclusion

We have introduced SINEX, a local data-agnostic post-hoc explainer for Siamese Networks able to process images, time-series and audio inputs in the context of *C*-way *k*-shot learning. By using a perturbation-based approach, both SINEX, and its coalition version SINEXC, are able to identify the important areas for SN classification, covering both positive and negative contributing features. SINEX enabled us to discover some of the limitations that SNs might encounter during their classification process, such as the significance of colors (on RGB images) or specific pixels areas (on grayscale images) that should not be considered important. Therefore, SINEX provides an effective tool to highlight such limitations and guide a subsequent model re-training phase. Finally, the proposed WPE and WNE metrics have allowed us to verify the independence of positive influencing segments and the

robustness of positive and negative segments on all datasets, regardless of both correct and incorrect classifications.

Future research directions will focus on various aspects. First, since a limitation of SINEX is that it can only study the SN behavior locally on each few-shot task, requiring human oversight in multiple analyses of different tasks to get a comprehensive understanding of the network's global behavior, inspired by Setzu et al. (2021) we aim at proposing a local-to-global abstraction of the logic learned by SN. Second, by combining SINEX explanations with WPE and WNE, we would like to identify those tasks and behavior that go against the verified segment dependence and might reveal miss-classifications. Third, we would like to leverage the level of the explanations by linking the influencing positive and negative segments on the support set with positive/negative segments on the query image. Finally, we plan to conduct an extrinsic interpretability evaluation of SINEX explanations through a human decision-making task driven by its explanations. This will help us objectively evaluate the effectiveness of our explanations.

Appendix A: Datasets

Omniglot (OGT) is a Grayscale image dataset containing 1623 different handwritten characters from 50 different alphabets, with each character drawn by 20 different people. "Background" alphabets are usually employed for training purpose, while "evaluation" alphabets are only used to test the model performance. In our study, background and evaluation alphabets were considered as a unique set, from which 40 training alphabets, 5 validation alphabets and 5 test alphabets were drawn randomly. For this dataset, we pursued a *character recognition* task.

The *Caltech-UCSD Birds 200* (CUB) dataset counts 200 different classes, each containing from 41 to 60 RGB image samples per class with varying dimensions between [120, 500] for height and [121, 500] for width. In our study, we decided to randomly pick a total of 60 classes by filtering on those containing at least 60 image samples per class (the maximum) to be consistent with the other datasets train/val/test set split. By doing so, we reduced the number of total records from 12k down to 3,6k. Therefore, for CUB we have again 50 classes used for training, while the remaining 10 were split equally between validation and test set. We resized each RGB image so to result in height and width dimensions of 224×224.

The *FiftyWords* (50W) dataset (Rath & Manmatha, 2003) consists of word outlines taken from the George Washington library, with each case representing a word and a series formed by taking the height profile of the word. We generated gray-scale images from each series by using a line-width that resembled a pixel density similar to the OGT data-samples, resulting in images dimension of 114×114. The 50 classes were divided as 40 for train, 5 for validation and 5 to test the model. We used this dataset to pursue a *word recognition* task.

The *AudioMNIST* (AST) dataset comprises 30k recordings of spoken digits (0–9) in English. Each digit is repeated 50 times by 60 different speakers (12 females and 48 males). We used this dataset to pursue a *speaker recognition* task creating three disjoint sets: the training set was composed of 50 classes, while the remaining 10 were divided equally between validation and test set. For the *ESC-50* (ESC) dataset, which consists of

2k annotated 5-second audio clips belonging to 50 different classes with 40 repetitions per class, we performed a *environmental audio classification* task. We used 40 classes for training, 5 for validation, and 5 for testing. Both datasets were pre-processed with `librosa`[4] to extract their log-mel spectrogram representation, which is commonly used for audio classification tasks (Hershey, 2017; Piczak, 2015). `AST` tracks were down-sampled to 41kHz and zero-padded, so to have vectors of equal length. Then, each track was converted using an FFT window size of 4096, hop length of 197 samples, and 224 mel-bands. The dimension of resulting spectrograms was of 224×224.

For the `ESC` dataset, the tracks were converted using a FFT window size of 2048, hop length of 512 samples, and 128 mel-bands. The spectrogram resulting dimension in this case was of of 128×431.

Appendix B: Siamese network architectures

In line with Koch et al. (2015); Honka (2019); Acconcjaioco (2020); Zhang (2019), we used a 2-branched SN architecture for `AST`, `ESC`, `OGT` and `50W` datasets, with slight variations among them. The overall structure consists of two convolution-based encoders, a distance layer, and a final output scoring layer. The two encoders share the same architecture and weights, which are updated simultaneously during training to result in identical embedding sub-networks. Each encoder is composed of different 2D-convolution blocks followed by a max-pooling layer, and a fully connected layer that converts the encoded-input to a 1-dimension form. The two encoded inputs are then compared using a distance layer, and a fully connected layer with a single unit uses a sigmoid activation function to calculate the similarity score. For `AST`, `ESC` and `OGT`, the encoders have *three* 2D-convolution blocks with 64, 32, 12 filters, and kernel sizes 5×5, 5×5, 3×3, respectively. The fully connected layer consists of 4096 units, and the distance function we used is the Euclidean distance for `AST` and `ESC`, and Manhattan distance for `OGT`. For `50W`, the architecture has four 2D-convolution blocks with 128, 64, 32, 16 filters, and kernel sizes of 3×3 per block. Like for the other networks, the fully connected layer has 4096 units, and the Euclidean distance is used.

For the `CUB` dataset, we implemented a 3-branched SN architecture in line with Hoffer (2015), which includes three encoding-branches that shares the same architecture and weights between them. Each encoder has *three* 2D-convolution blocks with 64, 32, 16 filters, and kernel sizes 3×3 for all blocks, and a final fully connected layer of 4096 units. The distance function in this case is the *Sum of Squares*. This 3SN architecture differs from the previously described 2SN networks in terms of its output and training phase requirements. Unlike the 2SNs, which directly output a similarity score between the two inputs in the range of [0, 1], the 3SN outputs two distances: one between the query input x and a positive sample $x_{=}$, and the other between x and a negative sample $x_{\neq}$. We trained 2SNs to recognize whether a given input pair belongs to the same class (label 1) or different ones (label 0) with the *Binary cross-entropy* loss function. In contrast, the 3SN was trained using the *Triplet Loss* (Schroff et al., 2015) with a margin of 0.5, with the goal of pushing dissimilar pairs farther from similar pairs by at least a margin of 0.5.

[4] `librosa`: https://librosa.org/.

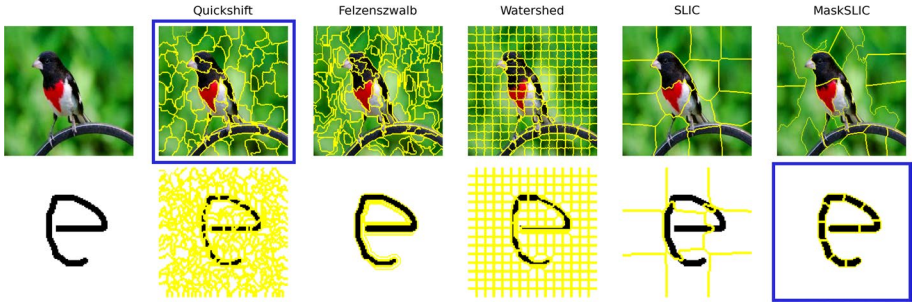

Fig. 9 Top row displays a sample from the CUB dataset alongside the corresponding segments obtained using different segmentation techniques from left to right: Quickshift, Felzenszwalb, Watershed, SLIC, and maskSLIC. The bottom row illustrates segments generated for a OGT dataset sample. Blue boxes indicate our chosen techniques for each of the two datasets. Parameters setting for the different algorithms are the following: *Quickshift*: {kernel size:1, max dist: 3, ratio: 0.5}; *Felzenszwalb*: {scale:1, sigma: 0.5, min size: 100}; *Watershed*: {markers:250, compactness: 0.01}; *SLIC*: {n segments:10, compactness: 100, sigma: 1, start label:1}; *MaskSLIC*: {n segments:10, compactness: 100, sigma: 1, start label:1 mask: {small objects min size: 200, small holes area threshold: 500}. Best viewed in colors (Color figure online)

We adopted the *Adam* optimizer to train all five networks. The SNs performances were evaluated every 100 training epochs on 300 random 5-way 1-shot tasks. For further details on how SNs are trained, including the ones used in this study, please refer to Koch et al. (2015); Wang (2020).

Appendix C: Segmentation algorithms

SINEX employs an internal segmentation algorithm to generate segments for subsequent perturbation. The segmentation algorithms supported by SINEX via the *scikit-image*[5] Python collection are: *Quickshift, Felzenszwalb, Watershed, SLIC*, and *maskSLIC*. We do not delve into the specific details of each algorithm here, as comprehensive descriptions are available in the official documentation. However, we outline the suggested approach for selecting the segmentation algorithm that best suits a given application. First and foremost, when selecting the algorithm, it is essential to strike a balance between (i) identifying segments that effectively capture the input features with semantic significance and (ii) ensuring that the total number of segments remains computationally manageable. The number of segments should be reasonable, allowing SINEX to compute efficiently within the available resources. We recommend beginning the process with a visual inspection of how different segmentation algorithms are applied to a given dataset. This visual inspection helps in gaining a better understanding of the algorithms and their suitability for each specific application. By visual inspection, it is possible to identify algorithms that may not be suitable or worthwhile for further investigation, allowing their exclusion from internal use within SINEX. In Fig. 9, we provide examples of the application of the various tested segmentation algorithms on both the CUB and OGT datasets. In our specific cases, we observed that *Quickshift* and *maskSLIC* performed better at segmenting areas of interest in the two datasets, respectively. However, we found that *Watershed* and *SLIC* tended to create equal square-like segments, failing to capture interesting pixel areas in

[5] https://scikit-image.org/.

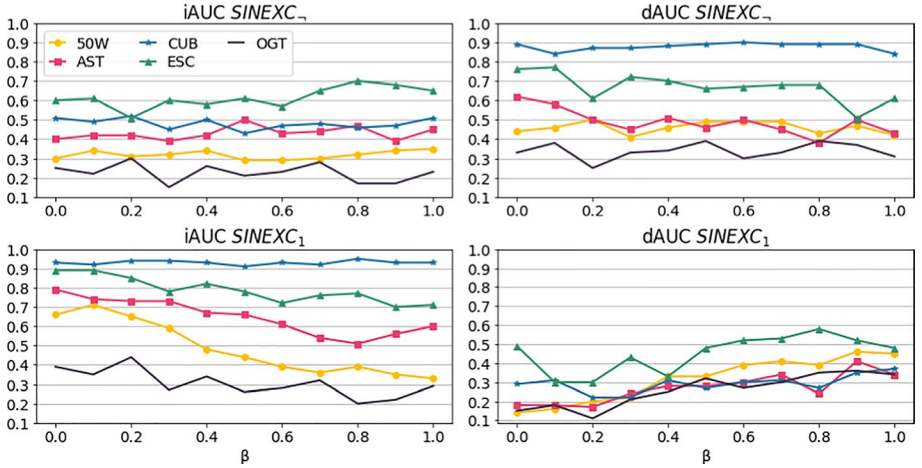

Fig. 10 Mean *iAUC* (on the left) and *dAUC* (on the right) values for SINEXC¬ (first row) and SINEXC_1 (second row), varying β per-coalition percentage of segments. Best viewed in colors (Color figure online)

our datasets. Therefore, we excluded them from our experiments on these datasets. Indeed, these algorithms are better suited for samples where distinct objects within the same image require clear separation, as suggested in the documentation.

Moreover, it is worth mentioning that each algorithm's performance can vary depending on different settings and configurations. Therefore, it is advisable to conduct various tests with different configurations before applying SINEX. It is also important to note that, in some cases, certain segmentation algorithms may be easily disregarded from this investigation as they fail to segment areas of interest, i.e., *Watershed* and *SLIC* in Fig. 9 for both `CUB` and `OGT`. We designed SINEX by drawing inspiration from established explainability techniques such as `LIME`, which also integrates segmentation algorithms with predefined settings. Consequently, we offer the flexibility to customize the choice of algorithm and its configuration to align with diverse application needs. As a starting point, we recommend selecting the algorithm through visual inspection.

In our experiments, we selected the following segmentation algorithms and set their additional parameters based on different preliminary experiments that allowed us to have a reasonable number of semantically meaningful segments to work with. For `AST` and `ESC`, we use the *Felzenszwalb* segmentation algorithm with a common setting of $sigma = 0.5$ and $scale = 1$ parameters. However, we use $min\_size = 600$ for `AST` and $min\_size = 900$ for `ESC`. The *MaskSLIC* algorithm segments `OGT` and `50W` datasets, with a common setting of $sigma = 1$ and $compactness = 100$, but varying with $n\_segments = 10$ for `OGT` and $n\_segments = 20$ for `50W`. For `CUB` dataset inputs, we use the *Quickshift* segmentation algorithm with $kernel\_size = 3$, $max\_dist = 60$, and $radio = 0.5$.

Appendix D: SINEXC per-coalition active segments analysis

To examine the impact of the β parameter on the performance of SINEXC, we conducted an experiment measuring *iAUC* and *dAUC* values while varying β in the range of [0.0, 1.0] with a step of 0.1. We recall that β represents the percentage of segments

Table 4 The table shows the best performing β for SINEXC$_\neg$ and SINEXC$_1$, along with the resulting *iAUC* and *dAUC* for each dataset

	SINEXC$_\neg$			SINEXC$_1$		
	β	*iAUC*	*dAUC*	β	*iAUC*	*dAUC*
OGT	0.2	0.3	0.25	0.2	0.44	0.11
CUB	0.1	0.51	0.85	0.3	0.96	0.22
50W	0.3	0.32	0.41	0.1	0.71	0.16
AST	0.5	0.5	0.46	0.2	0.73	0.17
ESC	0.9	0.68	0.51	0.1	0.89	0.30

that remain active or are "silenced" during the perturbation procedure, together with the segment under examination (see Sect. 4). For example, if $\beta = 0.2$ and $\omega = \neg$, each additional coalition of SINEXC will keep an additional 20% of random segments active with the segment under examination. On the other hand, if $\omega = 1$, each coalition will replace with non-informative values not only the segment in exam, but also an additional 20% of random selected other segments. We only considered 5-way 1-shot tasks with correct classifications for this experiment, and we kept the SINEXC α parameter at 200 for all experiments. Figure 10 shows the resulting *iAUC* and *dAUC* mean values for each dataset. Recalling that we want high *iAUC* and low *dAUC* simultaneously, our experiments show that the β parameter has not significant impact on the *iAUC* and *dAUC* values, except when paired with the $\omega = 1$ perturbation procedure. Indeed, SINEXC$_1$ curves expose an interesting behavior since increasing β value resulted in a decreasing *iAUC* trend and an increasing *dAUC* trend, simultaneously. This behavior is shared among all datasets, except CUB, which despite an outperforming *iAUC* score, its same curve seemed to be stable and did not seem to be influenced by the β parameter at all. However, the same behavior is not observed for SINEXC$_\neg$, which mostly results in poor *iAUC* and *dAUC*. The *iAUC* curves do not show any significant variation on different β values, and the *dAUC* curves only showed a slight decreasing trend for ESC and AST. Although the general performance for these two datasets is poor for high values of β, they perform better (lower *dAUC*) as we approach keeping the entire sample active during the perturbation procedure (i.e., close to $\beta = 1.0$).

Based on these findings, we identified the optimal β for each dataset and used them in experiments described in Sect. 5. Our selections, which prioritized a balance between high *iAUC* and low *dAUC*, are listed in Table 4. It is worth noting that the best β for SINEXC$_\neg$ span a wide range of [0.1, 0.9], while for SINEXC$_1$ they are much smaller, ranging only between [0.1, 0.3]. This suggests that β has less influence on the first perturbation procedure and that increasing the percentage of additional segments considered in each coalition may lead to worse performance for the second perturbation procedure (i.e., determining each segment contribution by considering the whole sample every time). This experiment, beside assessing β influence on SINEX, turned out to be a preliminary comparison of the two ω perturbation procedures as well. Our results show that SINEXC$_1$ is the better perturbation methodology, especially for the CUB, AST, ESC, and 50W datasets. In contrast, SINEXC$_\neg$ resulted in insertion and deletion curves that did not present an immediate raise or a instant drop respectively, which suggests that this method may not be able to accurately identify the important contributing segments. We suspect that SINEXC$_\neg$ tends to consider all segments to be equally important, since it deletes a significant amount of information at each perturbation step.

Appendix E: Additional qualitative evaluation

Examples of SINEX explanations for miss-classifications on the CUB, 50W, and AST datasets are illustrated in Fig. 11, 12 and 13 respectively. In Fig. 11, the *Rose breasted Grosbeak* sample (s_1) is miss-classified as a *Summer Tanager* sample s_3, due to the bird in s_1 being far and obscured by tree leaves, while the bird in s_3 is in the foreground. The explanation (h_3) shows that the SN is mainly relying on the red color, rather than learning to discriminate between the birds themselves. In Fig. 12, the miss-classification on 50W occurs because the query sample x is more similar to samples belonging to class $y = 5$ due to the presence of the up-down-up trend (positive segment in h_3). However, the s_1 support set sample, belonging to the same class as the query sample, presents a poor visible curve hump and a low final pick, which leads to its miss-classification as s_3. Finally, Fig. 13 shows a miss-classification example of woman generated recording (s_1;$y = 56$) as a male recorded audio

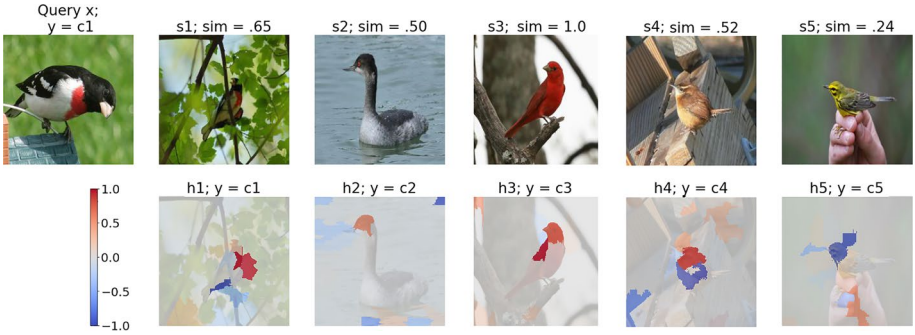

Fig. 11 SINEX$_1$ explanation of a miss-classified 5-way 1-shot task on the CUB dataset. c_1 (Rose breasted Grosbeak) is miss-classified for c_3 (Summer Tanager). Best viewed in colors (Color figure online)

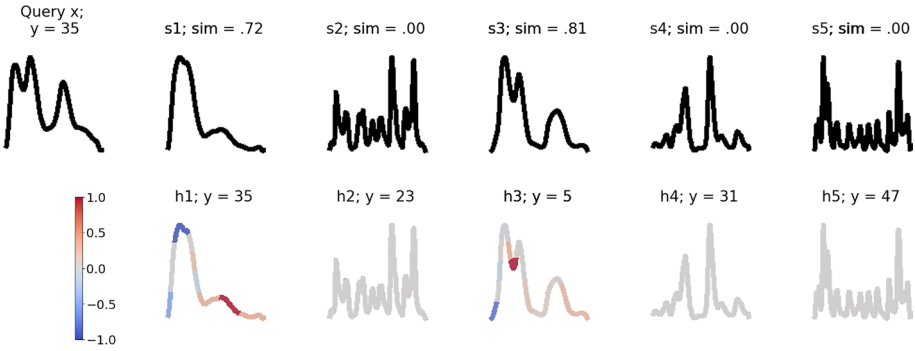

Fig. 12 SINEX$_1$ explanation on a 5-way 1-shot task on the 50W dataset. $y = 35$ is miss-classified for $y = 5$. Best viewed in colors (Color figure online)

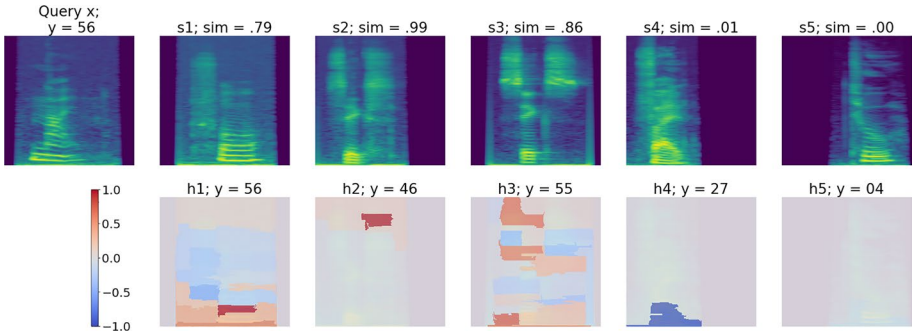

Fig. 13 SINEX$_1$ explanation of a miss-classified 5-way 1-shot task on the AST dataset. $y = 56$ is miss-classified for $y = 46$. Best viewed in colors (Color figure online)

(s_2;$y = 46$). The miss-classification occurred because the model relied too heavily on the lower frequency range in h_1 and placed too much importance on higher frequencies in h_2.

Appendix F: WPE-WNE analysis

The WPE and WNE analysis resulted in the same curve behavior for all dataset we experimented with, indicating that negative segments depend on positive ones, while the positives have a greater overall influence and are independent. In the cases where positive samples are independent the following results should be verified. The deletion of positive segments first results in a drop in the curve, while removing segments starting from negative ones does not result in such a drop ($dAUCN_{wpe} \gg dAUCP_{wpe}$). Additionally, removing positive segments results in a drop in the curve, even after negative segments have

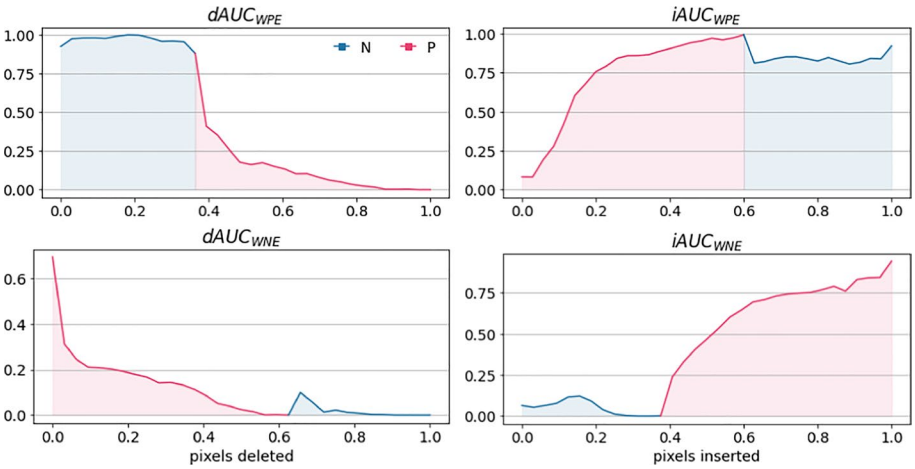

Fig. 14 Insertion (left) and deletion (right) curves for WPE (top row) and WNE (bottom row) procedures on correct and incorrect classifications in the 50W dataset. Blue colored portions indicate the removal or insertion of negative segments. Red colored areas represent positive segments. Best viewed in color (Color figure online)

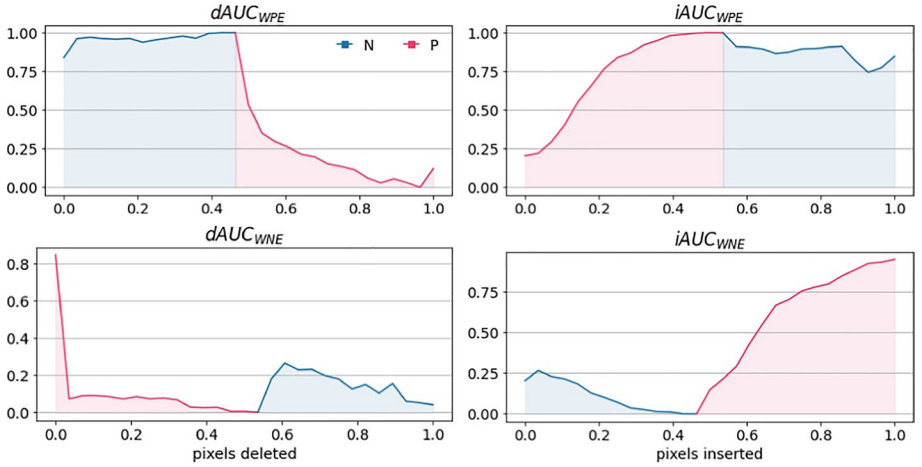

Fig. 15 Insertion (left) and deletion (right) curves for WPE (top row) and WNE (bottom row) procedures on correct and incorrect classifications in the AST dataset. Blue colored portions indicate the removal or insertion of negative segments. Red colored areas represent positive segments. Best viewed in color (Color figure online)

been removed, while removing negative segments alone keeps the curve small and stable ($dAUCP_{wpe} \simeq dAUCN_{wpe}$). Furthermore, the insertion of positive segments leads to an immediate raise in the curve compared to the insertion of negative segments into an empty sample ($iAUCP_{wpe} \gg iAUCN_{wpe}$). Finally, adding negative segments after positive ones maintains the curve at high similarity scores, which can be achieved by an immediate raise in the curve from the addition of positive segments even after negative ones have been inserted ($iAUCN_{wpe} \simeq iAUCP_{wpe}$). Figure 14 and 15 show the WPE and WNE curve for 50W and AST datasets respectively.

Appendix G: Shifted support sets analysis

To assess the influence of support set composition on both SN and SINEX performance, we report additional experiments comparing their performance with the standard support set against various shifted versions of the support set. We emphasize that we did not retrain the SNs on data augmented versions of the dataset, considering the shifted records. Instead, we conducted experiments using support sets with shifted records. The shifting procedure involves displacing the sample's content along the x and y axes. Specifically, we used $x = y = 20$, resulting in four shifting labels: $(-x, -y)$, $(x, -y)$, $(-x, y)$, and (x, y). These shifting operations in 50W, involve moving the actual drawn time-series from its original position to the top-left $(-x, -y)$, top-right $(x, -y)$, bottom-left $(-x, y)$, and bottom-right (x, y) areas of the image. Figure 16 illustrates the application of such shifting and the corresponding SINEX explanations for class label 23. Additional examples are provided for class 47 and 31 in Fig. 18 and Fig. 19, respectively. In general, SINEX explanations consistently highlight important contributing segments within the same pixel-areas of the drawn time-series, regardless of the applied shifting procedure. This consistency is evident when comparing non-shifted samples to $(-x, y)$ and (x, y) shifted samples in Fig. 16 and Fig. 18, as well as non-shifted samples to (x, y) shifted samples in Fig. 19. For these cases, SINEX

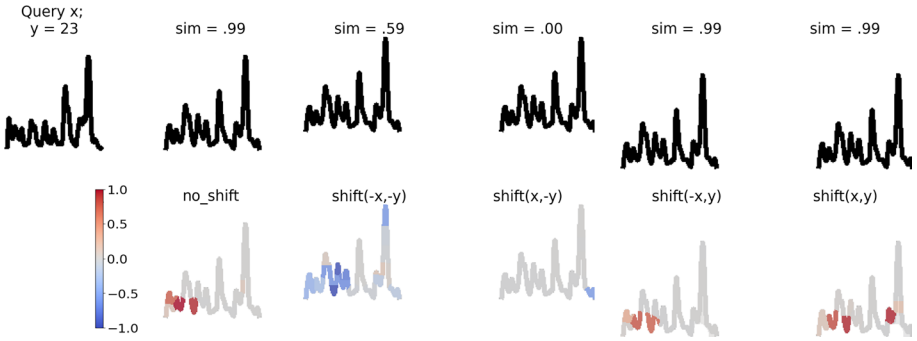

Fig. 16 Shifted support set examples in the `50W` dataset, class label 23. Top row consists of the query sample x followed by the original support set sample (no shift applied) belonging to the same class as x. Then from left to right the shifts are: $(-x, -y)$, $(x, -y)$, $(-x, y)$, and (x, y). The bottom row displays SINEX explanations for both the original and each shifted version of the support sample. Here, $x = y = 20$. This layout is repeated for all other shifted support sets examples in this section. Best viewed in color (Color figure online)

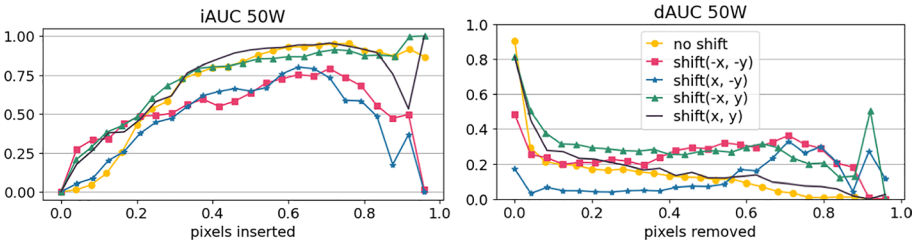

Fig. 17 Insertions (left) and deletions (right) curves for SINEX_1 on the `50W` dataset. Mean values across the 5 different test classes are compared between the original support set (no shift) and shifted support sets with the content shifted across the x and y axis. Here $x = y = 20$. Best viewed in colors (Color figure online)

Table 5 Mean *iAUC* and *dAUC* for SINEX_1 on the `50W` dataset

	iAUC	*dAUC*	Mean ± std
no shift	0.70	0.15	0.90 ± 0.07
$(-x, -y)$	0.52	0.24	0.48 ± 0.47
$(x, -y)$	0.46	0.12	0.17 ± 0.35
$(-x, y)$	0.69	0.28	0.81 ± 0.37
(x, y)	0.68	0.17	0.81 ± 0.37

The label "no shift" denotes the performance of the original support sets, while the following rows represent shifts in the content of the support set along the x and y axes. Here, $x = y = 20$. The table also presents the mean similarity score and the standard deviation between the query sample and the support set sample of the same class

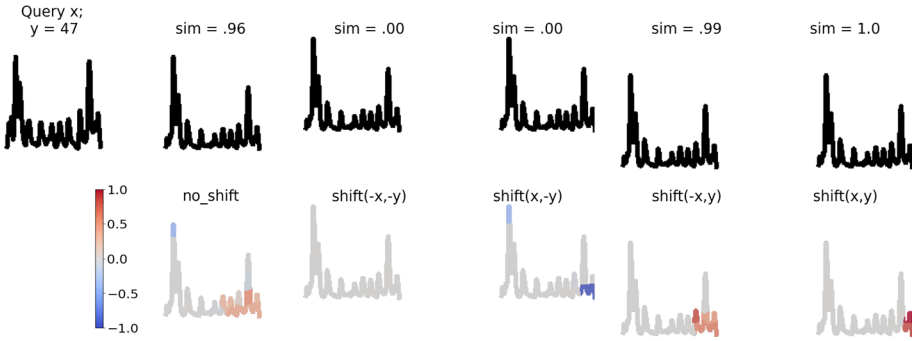

Fig. 18 Shifted support set example in the 50W dataset; class label 47. Here $x = y = 20$ (Color figure online)

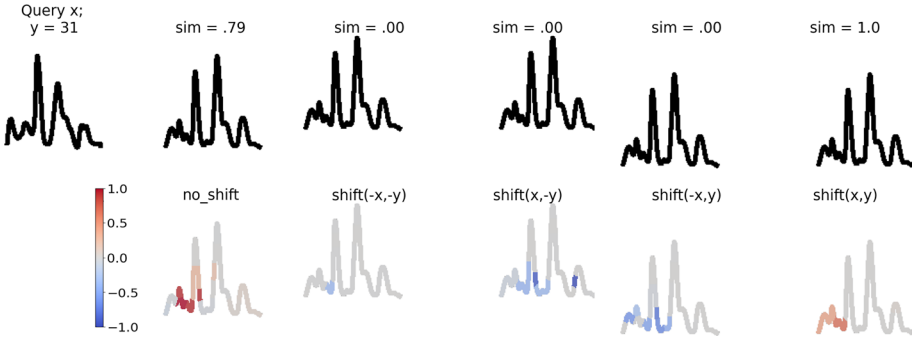

Fig. 19 Shifted support set example in the 50W dataset; class label 31. Here $x = y = 20$ (Color figure online)

identifies the same segments as positive contributions to the final similarity score, albeit with varying degrees of importance. However, when applying the $(-x, -y)$ and $(x, -y)$, they are sufficient to deceive the SN, while SINEX identifies these regions as negatively influencing. This behavior is consistent across all presented examples.

We present the quantitative evaluation results for 50W in Table 5, including mean *iAUC* and *dAUC*. We also calculate the mean similarity predicted score and standard deviation between the query sample x and the sample from the support set that belongs to the same class as x. The experimental setup mirrors the one used in the quantitative evaluation (Sect. 5), resulting in 150 5-way one-shot tasks per dataset, divided into 30 experiments for each of the 5 test classes. We compare the performance of each 5-way one-shot tasks using the original samples and after applying the four described different shifting processes to the support set samples. The mean insertion and deletion curves for this setting are illustrated in Fig. 17.

The best results are achieved on non-shifted support sets, characterized by high *iAUC* and low *dAUC* simultaneously. This indicates that the SN considers query samples x and support set samples from the same class to be very similar, and SINEX effectively identifies the important segments for similarity scoring. However, when support set samples are shifted, performance degrades. Regardless of the shift dimension, the mean

similarity score decreases, and the standard deviation increases significantly, ranging from 0.35 to 0.47. This suggests that the SN struggles with sample shifting, reinforcing the need for SN fine-tuning on data-augmented samples to ensure robust performance. The worst performing shifts are $(-x, -y)$ and $(x, -y)$ (negative shift on the y axis), which is reflected in their *iAUC* and *dAUC* curves. The *iAUC* curve for these negative shifts on the y-axis drops after 60% of pixels are inserted, reducing similarity scores to near zero. Also, the *dAUC* curves for these same negative shifts on the y-axis do not exhibit the expected drop; instead, they start with a low similarity mean score and show a slight increasing trend upon pixel removal. Conversely, the shifting procedures $(-x, y)$ and (x, y) yield mean similarity scores of 0.81, which closely approach the scores of non-shifted support sets. Although there is still a relatively high standard deviation, these results demonstrate the SN's ability to perform satisfactorily in these scenarios. This is also reflected in the SINEX *iAUC* and *dAUC* values. The insertion and deletion curves for these shifting procedures closely resemble those of the non-shifted support sets. This suggests that when the SN achieves robust classification, SINEX becomes a valuable tool for identifying the crucial segments contributing to these outcomes.

In conclusion, our analysis underscores the importance of additional fine-tuning for the SN or an enhanced training procedure that incorporates data-augmented samples. Specifically in `50W`, it is critical to train the network using samples shifted negatively along the y-axis. The utilization of SINEX provides valuable insights. Notably, the regions of interest for a query image and a support set sample remain consistent even when shifting is applied. This indicates the SN's effectiveness in identifying discriminative segments across unseen classes during training. SINEX explanations can be instrumental in guiding the generation of augmented samples for further fine-tuning or retraining. For instance, in `50W` it signals the need for fewer samples with positive y-axis shifts and more samples with negative y-axis shifts. This observation is primarily due to the decline in mean similarity scores, but it is reinforced by the fact that SINEX effectively identifies the segments that result in high *iAUC* and low *dAUC* values concurrently.

Acknowledgements This work has been partially supported by the European Community Horizon 2020 programme under the funding schemes: G.A. 871042 *SoBigData++*, G.A. 952026 *HumanE AI Net*, ERC-2018-ADG G.A. 834756 *XAI: Science and technology for the eXplanation of AI decision making*, G.A. 952215 *TAILOR*, and CHIST-ERA grant *CHIST-ERA-19-XAI-010*, by MUR (not yet avail.), FWF (I 5205), EPSRC (EP/V055712/1), NCN (2020/02/Y/ST6/00064), ETAg (SLTAT21096), BNSF (KP-06-AOO2/5) SAI. This work is partially supported by the European Union NextGenerationEU programme under the fuding schemes PNRR-PE-AI scheme (M4C2, investment 1.3, line on Artificial Intelligence) FAIR (Future Artificial Intelligence Research), and "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013.

Author contributions AF: Conceptualization, Methodology, Software, Validation, Investigation, Resources, Writing-Original Draft, Writing-Review & Editing, Visualization. RG: Conceptualization, Methodology, Investigation, Resources, Writing-Original Draft, Writing-Review & Editing, Supervision. DP: Conceptualization, Writing-Review & Editing, Supervision, Project administration.

Funding Open access funding provided by Università di Pisa within the CRUI-CARE Agreement. This work is supported by the European Community under the Horizon 2020 programme: G.A. 871042*SoBigData++, G.A. 952026HumanE AI Net, ERC-2018-ADG G.A. 834756XAI, G.A. 952,215TAILOR, CHIST-ERA grantCHIST-ERA-19-XAI-010SAI, FWF (I 5205), EPSRC (EP/V055712/1), NCN (2020/02/Y/ST6/00,064), ETAg (SLTAT21096), BNSF (KP-06-AOO2/5), and the NextGenerationEU programme under the funding schemes PNRR-PE-AI scheme (M4C2, investment 1.3, line on AI)FAIR(Future Artificial Intelligence Research), and "SoBigData.it- Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013.*

Availability of data and materials The open-source datasets adopted in this work are available at https://github.com/soerenab/AudioMNIST, https://github.com/karolpiczak/ESC-50, https://github.com/brendenlake/omniglot, http://www.timeseriesclassification.com/description.php?Dataset=FiftyWords, https://www.vision.caltech.edu/datasets/cub_200_2011/.

Code availability The code is open and available here: https://github.com/andreafedele/SINEX.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication The authors declare that they all provide consent for publication.

References

Acconcjaioco, M., et al. (2020). One-shot learning for acoustic identification of bird species in non-stationary environments. In *ICPR* (pp. 755–762). IEEE.

Achanta, R., et al. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*(11), 2274–2282.

Adadi, A., et al. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access, 6*, 52138–52160.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE, 10*(7), e0130140.

Beucher, S. (1992). The watershed transformation applied to image segmentation. *Scanning Microscopy, 1992*(6), 28.

Chen, C., et al. (2021). Self-learn to explain Siamese networks robustly. In *ICDM* (pp. 1018–1023). IEEE.

Dimitrova, D. (2020). The right to explanation under the right of access to personal data: Legal foundations in and beyond the gdpr. *The European Data Protection Law Review, 6*, 211.

Erhan, D., Courville, A., & Bengio, Y. (2010). Understanding representations learned in deep architectures.

Fedele, A., Guidotti, R., & Pedreschi, D. (2022). Explaining siamese networks in few-shot learning for audio data. In *DS, volume 13601 of Lecture Notes in Computer Science* (pp. 509–524). Springer.

Felzenszwalb, P. F., et al. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision, 59*(2), 167–181.

Fiaidhi, J., Mohammed, S., & Zezos, P. (2022). An xai thick data assisted caption generation for labeling severity of ulcerative colitis video colonoscopy. In *ICHI* (pp. 647–652). IEEE.

Flanagan, J. L. (2013). *Speech analysis synthesis and perception*, volume 3. Springer.

Fries, J., Wu, S., Ratner, A., & Ré, C. (2017). Swellshark: A generative model for biomedical named entity recognition without labeled data. arXiv:1704.06360.

Gong, X., Chen, X., Zhong, Z., & Chen, W. (2021). Enhanced few-shot learning for intrusion detection in railway video surveillance. *IEEE Transactions on Intelligent Transportation Systems*.

Guidotti, R., et al. (2019). Black box explanation by learning image exemplars in the latent feature space. In *ECML/PKDD, LNCS* (pp. 189–205). Springer.

Guidotti, R., et al. (2019). A survey of methods for explaining black box models. *ACM Computing Surveys, 51*(5), 93:1-93:42.

Gupta, P., Bhaskarpandit, S., & Gupta, M. (2021). Similarity learning based few shot learning for ECG time series classification. In *DICTA* (pp. 1–8). IEEE.
Haenlein, M., & Kaplan, A. (2019). A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review, 61*(4), 5–14.
Hershey, S., et al. (2017). CNN architectures for large-scale audio classification. In *ICASSP* (pp. 131–135). IEEE.
Hoffer, E., et al. (2015). Deep metric learning using triplet network. In *SIMBAD, volume 9370 of LNCS* (pp. 84–92). Springer.
Honka, T. (2019). One-shot learning with siamese networks for environmental audio.
Ienca, M., & Vayena, E. (2020). On the responsible use of digital data to tackle the covid-19 pandemic. *Nature Medicine, 26*(4), 463–464.
Irving, B. (2016). SLIC in a defined mask with applications to medical imaging. arxiv:abs/1606.09518
Iwata, T., & Kumagai, A. (2020). Few-shot learning for time-series forecasting. arXiv:2009.14379.
Jiang, L., Meng, D., Mitamura, T., & Hauptmann, A. G. (2014). Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 547–556).
Koch, G., Zemel, R., & Salakhutdinov, R., et al. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop, volume 2*. Lille.
Liu, B., Yu, X., Yu, A., Zhang, P., Wan, G., & Wang, R. (2019). Deep few-shot learning for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing, 57*(4), 2290–2304.
Looveren, A. V., et al. (2021). Interpretable counterfactual explanations guided by prototypes. In *ECML/PKDD, volume 12976 of LNCS* (pp. 650–665). Springer.
Lundberg, S. M., et al. (2017). A unified approach to interpreting model predictions. In *NIPS* (pp. 4765–4774).
Majid, S., Alenezi, F., Masood, S., Ahmad, M., Gündüz, E. S., & Polat, K. (2022). Attention based CNN model for fire detection and localization in real-world images. *Expert Systems with Applications, 189*, 116114.
Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence, 267*, 1–38.
Moujahid, H., Cherradi, B., Al-Sarem, M., Bahatti, L., Eljialy, A. B. A. M. Y., Alsaeedi, A., & Saeed, F. (2022). Combining cnn and grad-cam for covid-19 disease prediction and visual explanation. *Intelligent Automation & Soft Computing,32*(2).
Naudé, W. (2020). Artificial intelligence vs covid-19: limitations, constraints and pitfalls. *AI & society, 35*(3), 761–765.
Petsiuk, V., et al. (2018). RISE: randomized input sampling for explanation of black-box models. In *BMVC* (p. 151). BMVA Press.
Piczak, K. J. (2015). Environmental sound classification with convolutional neural networks. In *MLSP* (pp. 1–6). IEEE
Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S., & Sainath, T. N. (2019). Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing, 13*(2), 206–219.
Qiao, S., Liu, C., Shen, W., & Yuille, A. L. (2018). Few-shot image recognition by predicting parameters from activations. In *CVPR*.
Rahman, S., Khan, S. H., & Porikli, F. (2018). Zero-shot object detection: Learning to simultaneously recognize and localize novel concepts. In *ACCV (1), volume 11361 of Lecture Notes in Computer Science* (pp. 547–563). Springer.
Rath, T. M., & Manmatha, R. (2003). Word image matching using dynamic time warping. In *CVPR (2)* (pp. 521–527). IEEE Computer Society.
Ribeiro, M. T., et al. (2016). "Why Should I Trust You?": Explaining the predictions of any classifier. In *KDD* (pp. 1135–1144). ACM.
Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science, 2*(3), 160.
Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *CVPR* (pp. 815–823). IEEE Computer Society.
Selvaraju, R. R., et al. (2020). Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision, 128*(2), 336–359.
Setzu, M., Guidotti, R., Monreale, A., Turini, F., Pedreschi, D., & Giannotti, F. (2021). Glocalx - from local to global explanations of black box AI models. *Artificial Intelligence, 294*, 103457.
Snell, J., Swersky, K., & Zemel, R. S. (2017). Prototypical networks for few-shot learning. In *NIPS* (pp. 4077–4087).

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., & Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *CVPR* (pp. 1199–1208). Computer Vision Foundation/IEEE Computer Society.

Tummala, S., & Suresh, A. K. (2023). Few-shot learning using explainable siamese twin network for the automated classification of blood cells. Medical & Biological Engineering & Computing (pp. 1–15).

Utkin, L. V., et al. (2020). Explanation of siamese neural networks for weakly supervised learning. *Computer Informatics*, *39*(6).

Vedaldi, A., & Soatto, S. (2008). Quick shift and kernel methods for mode seeking. In *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part IV 10* (pp. 705–718). Springer.

Vélez, I. (2018). et al. One-shot speaker identification for a service robot using a cnn-based generic verifier. arxiv:abs/1809.04115.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching networks for one shot learning. In *NIPS* (pp. 3630–3638).

Wang, Y., et al. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM Computer Surveys, 53*(3), 63:1–63:34.

Ye, X., et al. (2020). Applying class-to-class SNs to explain classifications with supportive and contrastive cases. In *ICCBR, LNCS* (pp. 245–260). Springer.

Zeiler, M. D., et al. (2014). Visualizing and understanding convolutional networks. In *ECCV, volume 8689 of LNCS* (pp. 818–833). Springer.

Zhang, Y. et al. (2019). Siamese style convolutional neural networks for sound search by vocal imitation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (pp. 429–441).