A CAR Classifiers

In this appendix, we provide some details about our CAR classifiers.

Implementation. To determine the concept activation regions, we fit a SVC s_{κ}^{c} for each concept $c \in [C]$. This process is described in Algorithm 1.

Algorithm 1: Fit CAR Classifier

```
Input: Neural network $\boldsymbol{f} = \boldsymbol{l} \circ \boldsymbol{g} : \mathcal{X} \to \mathcal{H} \to \mathcal{Y}$, kernel function $\kappa : \mathcal{H}^2 \to \mathbb{R}^+$, concept
        positives $\mathcal{P}^c \subset \mathcal{X}$, concept negatives $\mathcal{N}^c \subset \mathcal{X}$
Output: CAR SVC Classifier $s_{\kappa}^{c} : \mathcal{H} \to \{0, 1\}$
$s_{\kappa}^{c} \leftarrow \mathrm{SVC}(\kappa)$ ;                                          /* Initialize SVC classifier */
$\mathcal{D} \leftarrow \{(\boldsymbol{g}(\boldsymbol{x}), \mathbb{I}_{\mathcal{P}^c}(\boldsymbol{x})) \mid \boldsymbol{x} \in \mathcal{P}^c \bigsqcup \mathcal{N}^c\}$ ;        /* Assemble SVC training set */
$(s_{\kappa}^{c}).\mathrm{fit}(\mathcal{D})$ ;                                               /* Fit SVC classifier */
return $s_{\kappa}^{c}$
```

where $\mathbb{I}_{\mathcal{P}^c}$ is the indicator function on the set $\mathcal{P}^c$. Our implementation leverages the SVC class from scikit-learn [71]. We use the default hyperparameters for each CAR classifier that appears in our experiments. The resulting classifier s_{κ}^{c} can then be used to assess if a test point $\boldsymbol{x} \in \mathcal{X}$ has a representation $\boldsymbol{g}(\boldsymbol{x}) \in \mathcal{H}$ that falls in the CAR $\mathcal{H}^c$ or not:

$$\boldsymbol{g}(\boldsymbol{x}) \in \begin{cases} \mathcal{H}^c & \text{if } s_{\kappa}^{c} \circ \boldsymbol{g}(\boldsymbol{x}) = 1 \\ \mathcal{H}^{\neg c} & \text{if } s_{\kappa}^{c} \circ \boldsymbol{g}(\boldsymbol{x}) = 0. \end{cases}$$

Choosing the kernel. Algorithm 1 requires the user to specify a kernel function κ. To select this kernel, a good approach is to train several SVCs s_{κ}^{c} with different kernels κ and see how each SVC generalizes on a validation set. We illustrate this process with MNIST in Figure 8. We note that the Gaussian RBF kernel outperforms the other kernels on the validation set, although the Matern kernel achieves perfect accuracy on the training set. This highlights the importance of evaluating a CAR classifier on a held-out dataset to make sure that the related CAR offers a good description of how concepts are distributed in the latent space $\mathcal{H}$. In our experiments, we found that Gaussian RBF kernels are often the most interesting option.

Concept set size. Algorithm 1 requires the user to specify concepts sets $\mathcal{P}^c$ and $\mathcal{N}^c$. What size $N^c = |\mathcal{P}^c| = |\mathcal{N}^c|$ should we choose for these concept sets? It seems logical that larger concepts sets are more likely to yield more accurate CARs. To study this experimentally, we propose to fit several concept classifiers for MNIST by varying the size N^c of their training concept sets $\mathcal{P}^c$ and

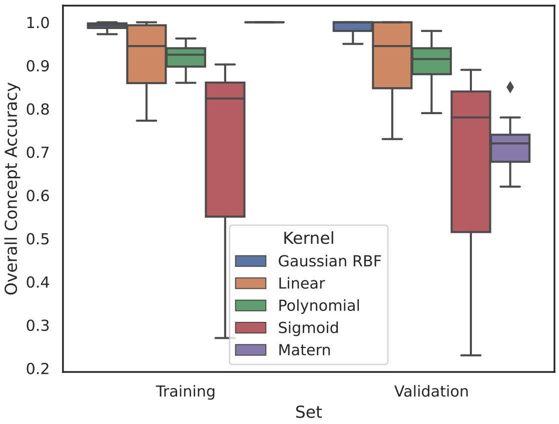


Figure 8: Accuracy of CAR classifiers for MNIST concepts with different kernels.

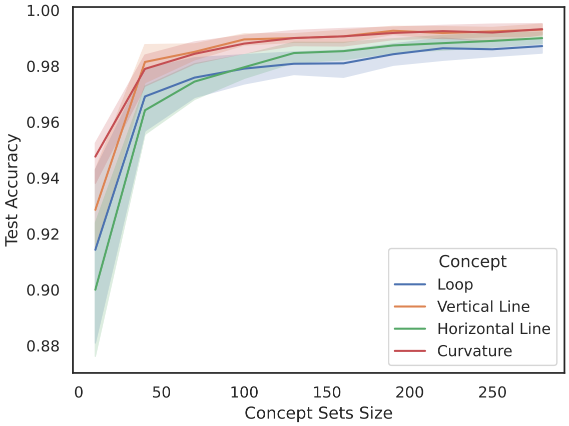


Figure 9: Accuracy of CAR classifiers for MNIST concepts for various concept set size N^c (average and 95% confidence intervals based on 10 runs).

$\mathcal{N}^c$. We report the results in Figure 9. As we can see, the curves flatten above $N^c > 200$. Increasing the concept sets size beyond this point does not improve the accuracy of the resulting CAR classifier. We recommend to acquire concept examples until the performance of the CAR classifier stabilizes. In our experiments, we found that $N^c = 200$ examples is often sufficient to obtain accurate CAR classifiers.

Tuning hyperparameters. In the case where the user desires a CAR classifier that generalizes as well as possible, tuning these hyperparameters might be useful. We propose to tune the kernel type, kernel width and error penalty of our CAR classifiers s^c_κ for each concept $c \in [C]$ by using Bayesian optimization and a validation concept set:

1. Randomly sample the hyperparameters from an initial prior distribution $\theta_h \sim P_{\text{prior}}$.
2. Split the concept sets $\mathcal{P}^c, \mathcal{N}^c$ into training concept sets $\mathcal{P}^c_{\text{train}}, \mathcal{N}^c_{\text{train}}$ and validation concept sets $\mathcal{P}^c_{\text{val}}, \mathcal{N}^c_{\text{val}}$.
3. For the current value θ_h of the hyperparameters, fit a model s^c_κ to discriminate the training concept sets $\mathcal{P}^c_{\text{train}}, \mathcal{N}^c_{\text{train}}$.
4. Measure the accuracy $\text{ACC}_{\text{val}} = \frac{\sum_{x\in\mathcal{P}^c_{\text{val}}} \mathbf{1}(s^c_\kappa \circ g(x)=1) + \sum_{x\in\mathcal{N}^c_{\text{val}}} \mathbf{1}(s^c_\kappa \circ g(x)=0)}{|\mathcal{P}^c_{\text{val}} \cup \mathcal{N}^c_{\text{val}}|}$.
5. Update the current hyperparameters θ_h based on ACC_{val} using Bayesian optimization (Optuna in our case).
6. Repeat 3-5 for a predetermined number of trials.

We applied this process to the CAR accuracy experiment (same setup as in Section 3.1.1 of the main paper) to tune the CAR classifiers for the CUB concepts. Interestingly, we noticed no improvement with respect to the CAR classifiers reported in the main paper: tuned and standard CAR classifier have an average accuracy of $(93 \pm .2)\%$ for the penultimate Inception layer. This suggests that the accuracy of CAR classifiers is not heavily dependant on hyperparameters in this case.

B TCAR Global Explanations

In this appendix, we provide some details about the TCAR scores.

Implementation. When the CAR classifiers are available, they permit to compute TCAR scores through Algorithm 2.

TCAR between concepts. Up until now, we have discussed TCAR scores that indicate how models relate classes to concepts. It is possible to define a similar score to estimate how models relate two

Algorithm 2: Compute Class-Concept TCAR Score

```
Input: Neural network $f = l \circ g : \mathcal{X} \to \mathcal{H} \to \mathcal{Y}$, CAR classifier $s^c_\kappa : \mathcal{H} \to \{0,1\}$, set of
        examples $\mathcal{D}_k \subset \mathcal{X}$ of a given class $k \in [d_Y]$
Output: TCAR score $\mathrm{TCAR}^c_k \in [0,1]$
count ← 0 ;                                        /* Initialize concept positive count */
for $x \in \mathcal{D}_k$ do
    if $s^c_\kappa \circ g(x) = 1$ then
        count ← count + 1 ;                /* Increment count if example in the CAR */
    end
end
$\mathrm{TCAR}^c_k$ ← count/$|\mathcal{D}_k|$ ;                     /* Normalize count to get TCAR */
return $TCAR^c_k$
```

concepts with each other. Given a set $\mathcal{D} \subset \mathcal{X}$ of examples, we define the TCAR score associated to the concepts $c_1, c_2 \in [C]$ as the ratio

$$\mathrm{TCAR}^{c_1,c_2} = \frac{|g(\mathcal{D}) \bigcap \mathcal{H}^{c_1} \bigcap \mathcal{H}^{c_2}|}{|g(\mathcal{D}) \bigcap (\mathcal{H}^{c_1} \bigcup \mathcal{H}^{c_2})|}.$$

Again, $\mathrm{TCAR} = 0$ corresponds to no overlap and $\mathrm{TCAR} = 1$ describes a perfect overlap. We note that the concept-concept TCAR score can be interpreted as a Jaccard index between the sets $g(\mathcal{D}) \bigcap \mathcal{H}^{c_1}$ and $g(\mathcal{D}) \bigcap \mathcal{H}^{c_2}$. This score is symmetric with respect to the concepts: $\mathrm{TCAR}^{c_1,c_2} = \mathrm{TCAR}^{c_2,c_1}$. The computation of this score is done as in Algorithm 3.

Algorithm 3: Compute Concept-Concept TCAR Score

```
Input: Neural network $f = l \circ g : \mathcal{X} \to \mathcal{H} \to \mathcal{Y}$, CAR classifiers $s^{c_1}_\kappa : \mathcal{H} \to \{0,1\}$ and
       $s^{c_2}_\kappa : \mathcal{H} \to \{0,1\}$, set of examples $\mathcal{D} \subset \mathcal{X}$
Output: TCAR score $\mathrm{TCAR}^{c_1,c_2} \in [0,1]$
numerator ← 0 ;                                        /* Initialize score numerator */
denominator ← 0 ;                                    /* Initialize score denominator */
for $x \in \mathcal{D}$ do
    if $s^{c_1}_\kappa \circ g(x) = 1$ and $s^{c_2}_\kappa \circ g(x) = 1$ then
        numerator ← numerator + 1 ;      /* Increment if example in both CARs */
    end
    if $s^{c_1}_\kappa \circ g(x) = 1$ or $s^{c_2}_\kappa \circ g(x) = 1$ then
        denominator ← denominator + 1 ;  /* Increment if example in one CAR */
    end
end
$\mathrm{TCAR}^{c_1,c_2}$ ← numerator/denominator ;                  /* Assemble the TCAR score */
return $TCAR^{c_1,c_2}$
```

TCAR between MNIST concepts. As an illustration, we compute concept-concept TCAR scores for the MNIST concepts and report the results in Figure 10. We see that the model relates concepts that tend to appear together (e.g. curvature and loop). Hence, concept-concept TCAR scores can serve as a proxy for the concept semantics encoded in a model's representation space. We note the similarity with the correlation between concept-based feature importance illustrated in Figure 6. The main difference is that concept-concept TCAR scores do not explicitly refer to input features.

Generalizing concept sensitivity. In our formalism, it is perfectly possible to define a *local* concept activation vector through the concept density $\rho^c : \mathcal{H} \to \mathbb{R}^+$ defined in Definition 2.1. Indeed, the vector $\nabla_{\boldsymbol{h}} \rho^c[\boldsymbol{h}] \in \mathcal{H}$ points in the direction of the representation space $\mathcal{H}$ where the concept density (and hence the presence of the concept) increases. Hence, this vector can be interpreted as a *local* concept activation vector. Note that this vector becomes global whenever we parametrize the concept density ρ^c with a linear kernel $\kappa(\boldsymbol{h}_1, \boldsymbol{h}_2) = \boldsymbol{h}_1^\intercal \boldsymbol{h}_2$. Equipped with this generalized notion of concept activation vector, we can also generalize the CAV concept sensitivity S^c_k by replacing the CAV w^c by $\nabla_{\boldsymbol{h}} \rho^c[\boldsymbol{h}]$ for the representation $\boldsymbol{h} = \boldsymbol{g}(\boldsymbol{x})$ of the input $\boldsymbol{x} \in \mathcal{X}$:

$$S^c_k(\boldsymbol{x}) \equiv (\nabla_{\boldsymbol{h}} \rho^c[\boldsymbol{g}(\boldsymbol{x})])^\intercal (\nabla_{\boldsymbol{h}} l_k[\boldsymbol{g}(\boldsymbol{x})]).$$

In this way, all the interpretation provided by the CAV formalism are also available in the CAR formalism.

C CAR Feature Importance

In this appendix, we provide some details about our concept-based feature importance.

Implementation. Concept-based feature importance uses the concept densities ρ^c to attribute an importance score to each feature in order to confirm/reject the presence of a concept $c \in [C]$ for a given example $\boldsymbol{x} \in \mathcal{X}$. This process is described in Algorithm 4. We stress that features importance scores are computed with *concept densities* and not with SVC classifiers. The reason for this is that SVCs are non-differentiable functions of the input and are therefore incompatible with gradient-based attribution methods such as Integrated-Gradients [25]. One could argue that a sparse version of the density ρ^c could be used by only allowing support vectors from the SVC to contribute. Our first implementation was relying on this approach. Unfortunately, this often led to vanishing importance scores. A possible explanation is the following: Gaussian radial basis function kernels $\kappa(\boldsymbol{h}_1, \boldsymbol{h}_2) = \exp\left[-(\gamma \left\| \boldsymbol{h}_1 - \boldsymbol{h}_2 \right\|_{\mathcal{H}})^2\right]$ decay very quickly as $\left\| \boldsymbol{h}_1 - \boldsymbol{h}_2 \right\|_{\mathcal{H}}$ increases. Hence, unless the example $\boldsymbol{x}$ we wish to explain has a representation $\boldsymbol{g}(\boldsymbol{x})$ located near a support vector of the SVC classifier, the sparse density (and its gradients) vanishes. On the other hand, using the density from Definition 2.1 allows us to incorporate the representations of all concept examples. This makes it more likely that the representation $\boldsymbol{g}(\boldsymbol{x})$ is located near (at least) one of the representation that contributes to the density. This permits to solve the vanishing gradient problem that led to vanishing attributions. It goes without saying that this solution comes at the expense of having to compute a density that scales linearly with the size N^c of the concept sets. In our implementation, we make this tractable by restricting to small concept sets ($N^c \leq 250$) and by saving the concept sets representations $\boldsymbol{g}(\mathcal{P}^c)$ and $\boldsymbol{g}(\mathcal{N}^c)$ to limit the number of queries to the model. We use Captum's implementation of feature importance methods [72]. When using radial basis function kernels, we tune the kernel width γ

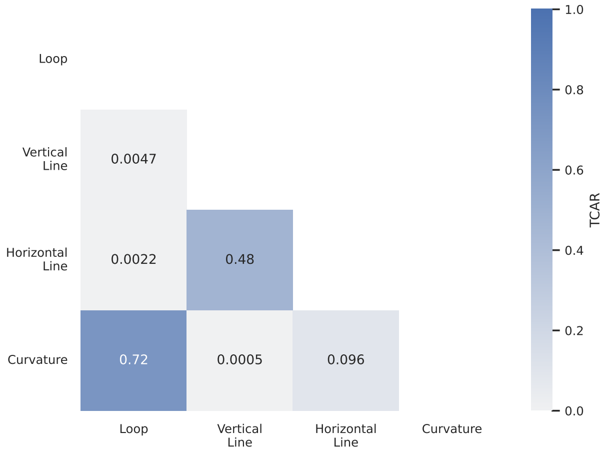


Figure 10: TCAR score between MNIST concepts.

with Optuna [73] with $1,000$ trials so that the Parzen window classifier $\mathbb{I}_{\mathbb{R}^+} \circ \rho^c$, where $\mathbb{I}_{\mathbb{R}^+}$ denotes the indicator function on $\mathbb{R}^+$, accurately discriminates between positives representations $\boldsymbol{g}(\mathcal{P}^c)$ and negative representations $\boldsymbol{g}(\mathcal{N}^c)$.

Algorithm 4: Compute concept-based feature importance

```
Input: Neural network $\boldsymbol{f} = \boldsymbol{l} \circ \boldsymbol{g} : \mathcal{X} \to \mathcal{H} \to \mathcal{Y}$, kernel function $\kappa : \mathcal{H}^2 \to \mathbb{R}^+$, concept
       positives $\mathcal{P}^c \subset \mathcal{X}$, concept negatives $\mathcal{N}^c \subset \mathcal{X}$, feature importance method
       $\boldsymbol{a} : \mathcal{A}(\mathbb{R}^{\mathcal{X}}) \times \mathcal{X} \to \mathbb{R}^{d_X}$, example $\boldsymbol{x} \in \mathcal{X}$
Output: Feature importance vector $\boldsymbol{a} \in \mathbb{R}^{d_X}$ for the example $\boldsymbol{x}$
$\rho^c \leftarrow \mathrm{Density}(\boldsymbol{g}, \kappa, \mathcal{P}^c, \mathcal{N}^c)$ ;        /* Initialize concept density as Def. 2.1 */
$\boldsymbol{a} \leftarrow \boldsymbol{a}(\rho^c \circ \boldsymbol{g}, \boldsymbol{x})$ ;                  /* Compute feature importance vector */
return $\boldsymbol{a}$
```

where $\mathcal{A}(\mathbb{R}^{\mathcal{X}})$ denotes the hypothesis set of scalar functions on the input space $\mathcal{X}$. In this case, this corresponds to the set of neural networks with input space $\mathcal{X}$ and scalar output.

Completeness. Completeness is a crucial property of some feature importance methods. It guarantees that the feature importance scores can be used to reconstruct the model prediction. To make this more precise, let $f : \mathcal{X} \to \mathbb{R}$ be a model with *scalar* output and let $\boldsymbol{x} \in \mathcal{X}$ be an example we wish to explain. Feature importance methods assign a score $a_i(f, \boldsymbol{x})$ to each feature $i \in [d_X]$. This score reflects the sensitivity of the model prediction $f(\boldsymbol{x})$ with respect to the component x_i of $\boldsymbol{x}$. Completeness is fulfilled whenever the sum of this importance scores equals the model prediction up to a constant baseline $b \in \mathbb{R}$: $\sum_{i=1}^{d_X} a_i(f, \boldsymbol{x}) = f(\boldsymbol{x}) - b$. The baseline varies from one method to the other. For instance, Lime [21] uses a vanishing baseline $b = 0$, SHAP [24] uses the average prediction $b = \mathbb{E}_{\boldsymbol{X}}\left[f(\boldsymbol{X})\right]$ and Integrated Gradients [25] use a baseline prediction $b = f(\bar{\boldsymbol{x}})$ for some baseline input $\bar{\boldsymbol{x}}$. With completeness, the importance scores $a_i(f, \boldsymbol{x})$ are given a natural interpretation: their sign indicates if the features tend to increase/decrease the prediction $f(\boldsymbol{x})$ and their absolute value indicates how important this effect is. When combined with CAR concept densities ρ^c, this interpretation is even more insightful.

Proposition C.1 (CAR Completeness)**.** *Consider a neural network decomposed as $\boldsymbol{f} = \boldsymbol{l} \circ \boldsymbol{g}$, where $\boldsymbol{g} : \mathcal{X} \to \mathcal{H}$ is a feature extractor mapping the input space $\mathcal{X}$ to the representation space $\mathcal{H}$ and $\boldsymbol{l}$ is a label function that maps the representation space $\mathcal{H}$ to the label set $\mathcal{Y}$. Let $\rho^c : \mathcal{H} \to \mathbb{R}^+$ be a concept density for some concept $c \in [C]$, defined as in Definition 2.1. Let $\boldsymbol{a} : \mathcal{A}(\mathbb{R}^{\mathcal{X}}) \times \mathcal{X} \to \mathbb{R}$ be a feature importance method satisfying the completeness property: $\sum_{i=1}^{d_X} a_i(f, \boldsymbol{x}) = f(\boldsymbol{x}) - b$ for some baseline $b \in \mathbb{R}$ and for all $\boldsymbol{x} \in \mathcal{X}$. Then, we have the following completeness property for the concept-based density:*

$$\sum_{i=1}^{d_X} a_i(\rho^c \circ \boldsymbol{g}, \boldsymbol{x}) = \rho^c \circ \boldsymbol{g}(\boldsymbol{x}) - b$$

Remark C.1. With this property, we can interpret features $i \in [d_X]$ with $a_i(\rho^c \circ \boldsymbol{g}, \boldsymbol{x}) > 0$ as those that tend to increase the concept density. This means that those features are important for the feature extractor $\boldsymbol{g}$ to map the example in a region of the representation space $\mathcal{H}$ where the concept is present. Hence, those are features that are important to identify a given concept $c \in [C]$. Conversely, features $i \in [d_X]$ with $a_i(\rho^c \circ \boldsymbol{g}, \boldsymbol{x}) < 0$ tend to decrease the concept density and therefore brings the example $\boldsymbol{x}$ in a region of the representation space $\mathcal{H}$ where the concept is absent. These features can therefore be interpreted as important to reject the presence of a given concept $c \in [C]$.

Proof. We simply note that $\rho^c \circ \boldsymbol{g}$ is a scalar function as $\rho^c \circ \boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^+$ for all $\boldsymbol{x} \in \mathcal{X}$. Hence, the proposition immediately follows by applying the completeness property to $f = \rho^c \circ \boldsymbol{g}$. □

Input baseline choice. The choice of baseline input $\bar{\boldsymbol{x}}$ has a notable effect on feature importance methods [74]. What constitutes a good input baseline is problem dependant. Intuitively, $\bar{\boldsymbol{x}}$ should correspond to an input $\boldsymbol{x} \in \mathcal{X}$ where no information is present [55]. Let us now explain how this information removal is achieved with the datasets that we use in our experiments. ① For MNIST, we chose a black image $\bar{\boldsymbol{x}} = \mathbf{0}$ as an input baseline. This is because MNIST images have a black background an the information comes from white pixels that represent the digits. ② For ECG, we chose a constant $\bar{\boldsymbol{x}} = \mathbf{0}$ time series as an input baseline. This is because ECG time series are normalized ($x_t \in [0, 1]$ for all $t \in [d_X]$) and the pulse information comes from time steps where the time series is non-vanishing $x_t \neq 0$. ③ For CUB, choosing a baseline is more complicated. The reason for this is that the background colour changes from one image to the other and rarely corresponds to a single colour. To address this, we proceed as in the literature [22] and select a baseline $\bar{\boldsymbol{x}}$ that corresponds to a blurred version of the image we wish to explain: $\bar{\boldsymbol{x}}(\boldsymbol{x}) = \boldsymbol{G}_\sigma \otimes \boldsymbol{x}$, where $\boldsymbol{G}_\sigma$ is a Gaussian filter of width σ and $\otimes$ denotes the convolution operation. We note that this baseline depends on which input $\boldsymbol{x} \in \mathcal{X}$ we want to explain. In our implementation, we use $\sigma = 50$ to have images that are significantly blurred. ④ For SEER, we chose a constant vector $\bar{\boldsymbol{x}} = \mathbf{0}$ as an input baseline. This is because all continuous features are standardized and all categorical features are one-hot encoded.

D CAR Latent Isometry Invariance

In this appendix, we prove that CAR explanations are invariant under isometries of the latent space when built with a radial kernel. Let us first rigorously define the notion of isometry between two vector spaces.

Definition D.1 (Isometry)**.** Let $(\mathcal{H}, \|\cdot\|_{\mathcal{H}})$ and $(\mathcal{H}', \|\cdot\|_{\mathcal{H}'})$ be two normed vector spaces. An *isometry* from $\mathcal{H}$ to $\mathcal{H}'$ is a map $\boldsymbol{\tau} : \mathcal{H} \rightarrow \mathcal{H}'$ such that for all $\boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathcal{H}$:

$$\|\boldsymbol{\tau}(\boldsymbol{h}_1) - \boldsymbol{\tau}(\boldsymbol{h}_2)\|_{\mathcal{H}'} = \|\boldsymbol{h}_1 - \boldsymbol{h}_2\|_{\mathcal{H}} .$$

We say that the two spaces $(\mathcal{H}, \|\cdot\|_{\mathcal{H}})$ and $(\mathcal{H}', \|\cdot\|_{\mathcal{H}'})$ are *isometric* if there exists a bijective isometry $\boldsymbol{\tau}$ from $\mathcal{H}$ to $\mathcal{H}'$.

An explanation method is invariant to latent space isometries if applying a bijective isometry $\boldsymbol{\tau}$ to the model's latent space $\mathcal{H}$ does not affect the explanations produced by the method. To make this more formal, we write the model as $\boldsymbol{f} = \boldsymbol{l} \circ \boldsymbol{g} = \boldsymbol{l} \circ \boldsymbol{\tau}^{-1} \circ \boldsymbol{\tau} \circ \boldsymbol{g}$, where $\boldsymbol{\tau}^{-1}$ is the inverse of the bijective isometry $\boldsymbol{\tau}$. In this setup, we could produce explanations with CARs by making the following replacements for the feature extractor and the label map: $\boldsymbol{g} \overset{\boldsymbol{\tau}}{\mapsto} \boldsymbol{\tau} \circ \boldsymbol{g}$ and $\boldsymbol{l} \overset{\boldsymbol{\tau}}{\mapsto} \boldsymbol{l} \circ \boldsymbol{\tau}^{-1}$. The explanations are defined to be invariant to latent space isometries if they are unaffected by this replacement. It is legitimate to expect this since the previous replacement leads to the same model $\boldsymbol{f} \overset{\boldsymbol{\tau}}{\mapsto} \boldsymbol{f}$ and substitutes the latent space by an isometric latent space $\mathcal{H} \overset{\boldsymbol{\tau}}{\mapsto} \mathcal{H}' = \boldsymbol{\tau}(\mathcal{H})$.

Let us now discuss the isometry invariance of CAR explanations. First, we recall that CARs are defined through a kernel κ. Not all kernels lead to isometry invariant CAR explanations. We will show that it holds for a family of kernels known as radial kernels.

Definition D.2 (Radial Kernel)**.** A *radial kernel* is a kernel function $\kappa : \mathcal{H}^2 \rightarrow \mathbb{R}^+$ that can be written as

$$\kappa(\boldsymbol{h}_1, \boldsymbol{h}_2) = \chi\left(\|\boldsymbol{h}_1 - \boldsymbol{h}_2\|_{\mathcal{H}}\right), \tag{1}$$

with a function $\chi : \mathbb{R}^+ \rightarrow \mathbb{R}^+$.

A typical example of radial kernel is the Gaussian radial basis function kernel (RBF) that corresponds to $\chi(x) = \exp\left[-(\gamma x)^2\right]$ for some $\gamma \in \mathbb{R}^+$. We are now ready to state and prove the isometry invariance property of CAR explanations.

Proposition D.1 (CAR Isometry Invariance)**.** *Consider a neural network decomposed as $\boldsymbol{f} = \boldsymbol{l} \circ \boldsymbol{g}$, where $\boldsymbol{g} : \mathcal{X} \rightarrow \mathcal{H}$ is a feature extractor mapping the input space $\mathcal{X}$ to the representation space $\mathcal{H}$ and $\boldsymbol{l}$ is a label function that maps the representation space $\mathcal{H}$ to the label set $\mathcal{Y}$. Let $\kappa : \mathcal{H}^2 \rightarrow \mathbb{R}^+$ be a radial kernel $\kappa(\boldsymbol{h}_1, \boldsymbol{h}_2) = \chi(\|\boldsymbol{h}_1 - \boldsymbol{h}_2\|_{\mathcal{H}})$ that we use to define CAR's concept density in Definition 2.1. Let $\boldsymbol{\tau} : \mathcal{H} \rightarrow \mathcal{H}'$ be a bijective isometry between the normed spaces $(\mathcal{H}, \|\cdot\|_{\mathcal{H}})$ and $(\mathcal{H}', \|\cdot\|_{\mathcal{H}'})$. All the explanations outputted by CAR remain the same if we transform the latent space with the isometry $\boldsymbol{\tau}$ by making the following replacements: $\boldsymbol{g} \overset{\boldsymbol{\tau}}{\mapsto} \boldsymbol{\tau} \circ \boldsymbol{g}$ and $\boldsymbol{l} \overset{\boldsymbol{\tau}}{\mapsto} \boldsymbol{l} \circ \boldsymbol{\tau}^{-1}$.*

Proof. For each concept $c \in [C]$, we note that CAR explanations exclusively rely on the concept density ρ^c from Definition 2.1 and the associated support vector classifier s^c_κ. Hence, it is sufficient to show that these two functions are invariant under isometry.

Concept Density. We start with the concept density ρ^c. We note that the kernel function κ can be applied to vectors from $\mathcal{H}'$ as $\kappa'(\boldsymbol{h}'_1, \boldsymbol{h}'_2) \equiv \chi(\left\|\boldsymbol{h}'_1 - \boldsymbol{h}'_2\right\|_{\mathcal{H}'})$ for all $(\boldsymbol{h}'_1, \boldsymbol{h}'_2) \in \mathcal{H}'$. Applying CAR in the latent space $\mathcal{H}'$ isometric to $\mathcal{H}$ corresponds to using this kernel to compute an alternative density ρ'^c. Let us fix an example $\boldsymbol{x} \in \mathcal{X}$. Under the isometry, the concept density for this example transforms as $\rho^c \circ \boldsymbol{g}(\boldsymbol{x}) \overset{\boldsymbol{\tau}}{\mapsto} \rho'^c \circ \boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x})$. Let us show that this is in fact an invariance:

$$\begin{aligned}
\rho'^c \circ \boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}) &\overset{\text{Def. 2.1}}{=} \sum_{n=1}^{N^c} \kappa'\left[\boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}^{c,n})\right] - \kappa'\left[\boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}^{\neg c,n})\right] \\
&\overset{\text{Def. D.2}}{=} \sum_{n=1}^{N^c} \chi\left(\|\boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}^{c,n})\|_{\mathcal{H}'}\right) - \chi\left(\|\boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}^{\neg c,n})\|_{\mathcal{H}'}\right) \\
&\overset{\text{Def. D.1}}{=} \sum_{n=1}^{N^c} \chi\left(\|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}^{c,n})\|_{\mathcal{H}}\right) - \chi\left(\|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}^{\neg c,n})\|_{\mathcal{H}}\right) \\
&\overset{\text{Def. D.2}}{=} \sum_{n=1}^{N^c} \kappa\left[\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x}^{c,n})\right] - \kappa\left[\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x}^{\neg c,n})\right] \\
&\overset{\text{Def. 2.1}}{=} \rho^c \circ \boldsymbol{g}(\boldsymbol{x}).
\end{aligned}$$

We deduce that the concept density is invariant under isometry: $\rho^c \circ \boldsymbol{g}(\boldsymbol{x}) \overset{\boldsymbol{\tau}}{\mapsto} \rho^c \circ \boldsymbol{g}(\boldsymbol{x})$.

SVC. The proof is more involved for the SVC s^c_κ. We assume that the reader is familiar with the standard theory of SVC. If this is not the case, please refer e.g. to Chapter 7 of [75]. For the sake of notation, we will abbreviate $\boldsymbol{g}(\boldsymbol{x}^{c,n})$ and $\boldsymbol{g}(\boldsymbol{x}^{\neg c,n})$ by $\boldsymbol{h}^{c,n}$ and $\boldsymbol{h}^{\neg c,n}$ respectively. Similarly, we abbreviate $\boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}^{c,n})$ and $\boldsymbol{\tau} \circ \boldsymbol{g}(\boldsymbol{x}^{\neg c,n})$ by $\boldsymbol{h}'^{c,n}$ and $\boldsymbol{h}'^{\neg c,n}$ respectively. The SVC concept classifier can be written as

$$s^c_\kappa(\boldsymbol{h}) = \mathbb{I}_{\mathbb{R}^+}\left(\sum_{n=1}^{N^c} \alpha^{c,n} \kappa[\boldsymbol{h}, \boldsymbol{h}^{c,n}] - \alpha^{\neg c,n} \kappa[\boldsymbol{h}, \boldsymbol{h}^{\neg c,n}] + \beta\right), \tag{2}$$

where $\mathbb{I}_{\mathbb{R}^+}$ is the indicator function on $\mathbb{R}^+$, the real numbers $\alpha^{c,n}, \alpha^{\neg c,n}$ maximize the objective

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\alpha}^c, \boldsymbol{\alpha}^{\neg c}) = &\sum_{n=1}^{N^c} \alpha^{c,n} + \alpha^{\neg c,n} - \frac{1}{2} \sum_{n=1}^{N^c} \sum_{m=1}^{N^c} \alpha^{c,n} \alpha^{c,m} \kappa\left[\boldsymbol{h}^{c,n}, \boldsymbol{h}^{c,m}\right] \\
&+ \alpha^{\neg c,n} \alpha^{\neg c,m} \kappa\left[\boldsymbol{h}^{\neg c,n}, \boldsymbol{h}^{\neg c,m}\right] - 2\alpha^{c,n} \alpha^{\neg c,m} \kappa\left[\boldsymbol{h}^{c,n}, \boldsymbol{h}^{\neg c,m}\right]
\end{aligned} \tag{3}$$

under the constraints

$$\begin{aligned}
&\alpha^{c,n} \geq 0 \qquad \alpha^{\neg c,n} \geq 0 \qquad \forall n \in [N^c], \\
&\sum_{n=1}^{N^c} \alpha^{c,n} - \alpha^{\neg c,n} = 0.
\end{aligned}$$

Finally, the bias term can be written as

$$\begin{aligned}
\beta = &\frac{1}{|\mathcal{S}^c| + |\mathcal{S}^{\neg c}|} \Bigg(|\mathcal{S}^c| - |\mathcal{S}^{\neg c}| - \sum_{n \in \mathcal{S}^c} \sum_{m \in \mathcal{S}^c} \alpha^{c,m} \kappa\left[\boldsymbol{h}^{c,n}, \boldsymbol{h}^{c,m}\right] \\
&+ \sum_{n \in \mathcal{S}^c} \sum_{m \in \mathcal{S}^{\neg c}} \alpha^{\neg c,m} \kappa\left[\boldsymbol{h}^{c,n}, \boldsymbol{h}^{\neg c,m}\right] - \sum_{n \in \mathcal{S}^{\neg c}} \sum_{m \in \mathcal{S}^c} \alpha^{c,m} \kappa\left[\boldsymbol{h}^{\neg c,n}, \boldsymbol{h}^{c,m}\right] \\
&+ \sum_{n \in \mathcal{S}^{\neg c}} \sum_{m \in \mathcal{S}^{\neg c}} \alpha^{\neg c,m} \kappa\left[\boldsymbol{h}^{\neg c,n}, \boldsymbol{h}^{\neg c,m}\right] \Bigg),
\end{aligned} \tag{4}$$

where $\mathcal{S}^c = \{n \in [N^c] \mid \alpha^{c,n} \neq 0\}$ and $\mathcal{S}^{\neg c} = \{n \in [N^c] \mid \alpha^{\neg c,n} \neq 0\}$ are the indices of the support vectors. Under isometry, the SVC classification for a latent vector $\boldsymbol{h} \in \mathcal{H}$ transforms as $s^c_\kappa(\boldsymbol{h}) \overset{\boldsymbol{\tau}}{\mapsto} s'^c_{\kappa'}(\boldsymbol{h}')$ with $\boldsymbol{h}' = \boldsymbol{\tau}(\boldsymbol{h})$ and

$$s'^c_{\kappa'}(\boldsymbol{h}') = \mathbb{I}_{\mathbb{R}^+}\left(\sum_{n=1}^{N^c} \underbrace{\alpha'^{c,n}}_{②} \underbrace{\kappa'[\boldsymbol{h}', \boldsymbol{h}'^{c,n}]}_{①} - \underbrace{\alpha'^{\neg c,n}}_{②} \underbrace{\kappa'[\boldsymbol{h}', \boldsymbol{h}'^{\neg c,n}]}_{①} + \underbrace{\beta'}_{③}\right). \tag{5}$$

We will now show that $s^c_\kappa(\boldsymbol{h}) = s'^c_{\kappa'}(\boldsymbol{h}')$ so that the SVC is invariant under isometries. We proceed in 3 steps.

① We show that $\kappa'\left[\boldsymbol{h}'_1, \boldsymbol{h}'_2\right] = \kappa\left[\boldsymbol{h}_1, \boldsymbol{h}_2\right]$ for any $(\boldsymbol{h}_1, \boldsymbol{h}_2) \in \mathcal{H}^2$ and $\boldsymbol{h}'_1 = \boldsymbol{\tau}(\boldsymbol{h}_1), \boldsymbol{h}'_2 = \boldsymbol{\tau}(\boldsymbol{h}_2)$:

$$\begin{aligned}
\kappa'\left[\boldsymbol{h}'_1, \boldsymbol{h}'_2\right] &\overset{\text{Def. D.2}}{=} \chi\left(\|\boldsymbol{\tau}(\boldsymbol{h}_1) - \boldsymbol{\tau}(\boldsymbol{h}_2)\|_{\mathcal{H}'}\right) \\
&\overset{\text{Def. D.1}}{=} \chi\left(\|\boldsymbol{h}_1 - \boldsymbol{h}_2\|_{\mathcal{H}}\right) \\
&\overset{\text{Def. D.2}}{=} \kappa\left[\boldsymbol{h}_1, \boldsymbol{h}_2\right].
\end{aligned}$$

By injecting this in (5), we are able to make the following replacements: $\kappa'\left[\boldsymbol{h}', \boldsymbol{h}'^{c,n}\right] = \kappa\left[\boldsymbol{h}, \boldsymbol{h}^{c,n}\right]$ and $\kappa'\left[\boldsymbol{h}', \boldsymbol{h}'^{\neg c,n}\right] = \kappa\left[\boldsymbol{h}, \boldsymbol{h}^{\neg c,n}\right]$ for all $n \in [N^c]$.

② We show that $\alpha'^{c,n} = \alpha^{c,n}$ and $\alpha'^{\neg c,n} = \alpha^{\neg c,n}$ for all $n \in [N^c]$. To that aim, we note that $\boldsymbol{\alpha}'^c$ and $\boldsymbol{\alpha}'^{\neg c}$ maximize the objective

$$\begin{aligned}
\mathcal{L}'(\boldsymbol{\alpha}'^c, \boldsymbol{\alpha}'^{\neg c}) =& \sum_{n=1}^{N^c} \alpha'^{c,n} + \alpha'^{\neg c,n} - \frac{1}{2} \sum_{n=1}^{N^c} \sum_{m=1}^{N^c} \alpha'^{c,n} \alpha'^{c,m} \kappa'\left[\boldsymbol{h}'^{c,n}, \boldsymbol{h}'^{c,m}\right] \\
&+ \alpha'^{\neg c,n} \alpha'^{\neg c,m} \kappa'\left[\boldsymbol{h}'^{\neg c,n}, \boldsymbol{h}'^{\neg c,m}\right] - 2\alpha'^{c,n} \alpha'^{\neg c,m} \kappa'\left[\boldsymbol{h}'^{c,n}, \boldsymbol{h}'^{\neg c,m}\right] \\
\overset{①}{=}& \sum_{n=1}^{N^c} \alpha'^{c,n} + \alpha'^{\neg c,n} - \frac{1}{2} \sum_{n=1}^{N^c} \sum_{m=1}^{N^c} \alpha'^{c,n} \alpha'^{c,m} \kappa\left[\boldsymbol{h}^{c,n}, \boldsymbol{h}^{c,m}\right] \\
&+ \alpha'^{\neg c,n} \alpha'^{\neg c,m} \kappa\left[\boldsymbol{h}^{\neg c,n}, \boldsymbol{h}^{\neg c,m}\right] - 2\alpha'^{c,n} \alpha'^{\neg c,m} \kappa\left[\boldsymbol{h}^{c,n}, \boldsymbol{h}^{\neg c,m}\right] \\
\overset{(3)}{=}& \mathcal{L}(\boldsymbol{\alpha}'^c, \boldsymbol{\alpha}'^{\neg c}).
\end{aligned}$$

Since this objective is identical to the one from the original SVC and the constraints are unaffected by the isometry, we deduce that the solution to this convex optimization problem is identical to the solution of (3). Hence, we have that $\alpha'^{c,n} = \alpha^{c,n}$ and $\alpha'^{\neg c,n} = \alpha^{\neg c,n}$ for all $n \in [N^c]$. Again, we can make these replacements in (5).

③ We show that $\beta' = \beta$. First, we note that the support vector indices are invariant under isometry: $\mathcal{S}'^c = \{n \in [N^c] \mid \alpha'^{c,n} \neq 0\} = \{n \in [N^c] \mid \alpha^{c,n} \neq 0\} = \mathcal{S}^c$ and similarly $\mathcal{S}'^{\neg c} = \mathcal{S}^{\neg c}$. Hence we have

$$\begin{aligned}
\beta' =& \frac{1}{|\mathcal{S}^c| + |\mathcal{S}^{\neg c}|} \Bigg(|\mathcal{S}^c| - |\mathcal{S}^{\neg c}| - \sum_{n \in \mathcal{S}^c} \sum_{m \in \mathcal{S}^c} \alpha'^{c,m} \kappa'\left[\boldsymbol{h}'^{c,n}, \boldsymbol{h}'^{c,m}\right] \\
&+ \sum_{n \in \mathcal{S}^c} \sum_{m \in \mathcal{S}^{\neg c}} \alpha'^{\neg c,m} \kappa'\left[\boldsymbol{h}'^{c,n}, \boldsymbol{h}'^{\neg c,m}\right] - \sum_{n \in \mathcal{S}^{\neg c}} \sum_{m \in \mathcal{S}^c} \alpha'^{c,m} \kappa'\left[\boldsymbol{h}'^{\neg c,n}, \boldsymbol{h}'^{c,m}\right] \\
&+ \sum_{n \in \mathcal{S}^{\neg c}} \sum_{m \in \mathcal{S}^{\neg c}} \alpha'^{\neg c,m} \kappa'\left[\boldsymbol{h}'^{\neg c,n}, \boldsymbol{h}'^{\neg c,m}\right] \Bigg) \\
\overset{①\&②}{=}& \frac{1}{|\mathcal{S}^c| + |\mathcal{S}^{\neg c}|} \Bigg(|\mathcal{S}^c| - |\mathcal{S}^{\neg c}| - \sum_{n \in \mathcal{S}^c} \sum_{m \in \mathcal{S}^c} \alpha^{c,m} \kappa\left[\boldsymbol{h}^{c,n}, \boldsymbol{h}^{c,m}\right] \\
&+ \sum_{n \in \mathcal{S}^c} \sum_{m \in \mathcal{S}^{\neg c}} \alpha^{\neg c,m} \kappa\left[\boldsymbol{h}^{c,n}, \boldsymbol{h}^{\neg c,m}\right] - \sum_{n \in \mathcal{S}^{\neg c}} \sum_{m \in \mathcal{S}^c} \alpha^{c,m} \kappa\left[\boldsymbol{h}^{\neg c,n}, \boldsymbol{h}^{c,m}\right] \\
&+ \sum_{n \in \mathcal{S}^{\neg c}} \sum_{m \in \mathcal{S}^{\neg c}} \alpha^{\neg c,m} \kappa\left[\boldsymbol{h}^{\neg c,n}, \boldsymbol{h}^{\neg c,m}\right] \Bigg) \\
=& \beta.
\end{aligned}$$

By making the replacements from points ①, ② and ③ in (5), we deduce that $s^c_\kappa(\boldsymbol{h}) = s'^c_{\kappa'}(\boldsymbol{h}')$. □

E Empirical Evaluation

This appendix provides useful details to reproduce the empirical evaluation from Section 3.

Computing Resources. All the empirical evaluations were run on a single machine equipped with a 18-Core Intel Core i9-10980XE CPU and a NVIDIA RTX A4000 GPU. The machine runs on Python 3.9 [76] and Pytorch 1.10.2 [77].

Dataset licenses. The MNIST dataset dataset is made available under the terms of the Creative Commons Attribution-Share Alike 3.0 License. The ECG dataset dataset is made available under the terms of the Open Data Commons Attribution License v1.0. The CUB dataset is made available for non-commercial research and educational purposes.

Models. The detailed architecture of the models are provided in Tables 2, 3 and 4. The InceptionV3 architecture is the same as in the literature [60]. We use its official Pytorch implementation.

Data Split. All the datasets are naturally split in training and testing data. In the ECG dataset, the different types of abnormal heartbeats are imbalanced (e.g. the fusion beats constitute only 0.7% of the training set). Hence, we create a synthetic training set with balanced concepts using SMOTE [78]. As in [41], the CUB dataset is augmented by using random crops and random horizontal flips.

Model Fitting. In fitting each model, we use the test set as a validation set since our purpose is not to obtain the models with the best generalization but simply models that perform well on a set of examples we wish to explain (here the examples from the test set). All the models are trained to minimize the cross-entropy between their prediction and the true labels. The hyperparameters are as follows. ① For MNIST we use a Adam optimizer with batches of 120 examples, a learning rate of 10^{-3}, a weight decay of 10^{-5} for 50 epochs with patience 10. ② For ECG we use a Adam optimizer with batches of 300 examples, a learning rate of 10^{-3}, a weight decay of 10^{-5} for 50 epochs with patience 10. ③ For CUB, we use a stochastic gradient descent optimizer with batches of 64 examples, a learning rate of 10^{-3}, a weight decay of $4 \cdot 10^{-5}$ for $1,000$ epochs with patience 50.

Concepts. The concept mapping between MNIST classes and concepts is provided in Table 5. For the ECG and the CUB datasets, the presence/absence of a concept for each example is readily available in the dataset.

Concept classifiers. All the concept classifiers are implemented with scikit-learn [71]. For CAR classifiers, we fit a SVC with Gaussian RBF kernel and default hyperparameters from scikit-learn. For CAV classifiers, we fit a linear classifier with a stochastic gradient descent optimizer with learning rate 10^{-2} and a tolerance of 10^{-3} for $1,000$ epochs and the remaining default hyperparameters from scikit-learn.

Statistical significance. The statistical significance test from Section 3.1.1 is performed with the scikit-learn [71] implementation of the permutation test. For MNIST and ECG, we consider 100 permutations per concept. For CUB, this test is more expensive since the latent spaces are high-dimensional. We consider only 25 permutations per concept in that case.

Concept-based feature importance. In the experiment from Section 2.3, we use Captum's implementation [72] of Integrated Gradients [25] with default parameters. In the case of CUB, storing the feature importance scores for each concept and for the whole test set requires a prohibitive amount of memory. To avoid this problem, we select $C = 6$ concepts and subsample 50 positive and 50 negative

Table 2: MNIST Model

Block Name	Layer Type	Hyperparameters	Activation
Conv1	Conv2d Dropout2d MaxPool2d	Input Channels = 1, Output Channels = 16, Kernel Size = 5, Stride = 1, Padding = 0 p = 0.2 Kernel Size = 2, Stride = 2	ReLU
Conv2	Conv2d Dropout2d MaxPool2d	Input Channels = 16, Output Channels = 32, Kernel Size = 5, Stride = 1, Padding = 0 p = 0.2 Kernel Size = 2, Stride = 2	ReLU
Lin1	Flatten Linear	 Input Features = 512, Output Features = 10	
Lin2	Dropout Linear	p = 0.2 Input Features = 10, Output Features = 5	
	Dropout Linear	p = 0.2 Input Features = 5, Output Features = 10	

Table 3: ECG Model

Block Name	Layer Type	Hyperparameters	Activation
Conv1	Conv1d MaxPool1d	Input Channels = 1, Output Channels = 16, Kernel Size = 3, Stride = 1, Padding = 1 Kernel Size = 2	
Conv2	Conv1d MaxPool1d	Input Channels = 16, Output Channels = 64, Kernel Size = 3, Stride = 1, Padding = 1 Kernel Size = 2	
Conv3	Conv1d MaxPool1d	Input Channels = 64, Output Channels = 128, Kernel Size = 3, Stride = 1, Padding = 1 Kernel Size = 2	
Lin	Flatten Linear	Input Features = 2944, Output Features = 32	
	Leaky ReLU Linear	Negative Slope = 10^{-2} Input Features = 32, Output Features = 2	Leaky ReLU

Table 4: CUB Model

Block Name	Layer Type	Hyperparameters	Activation
InceptionOut	InceptionV3 [60]	Pretrained = True	
	Linear	Input Features = 2048, Output Features = 200	

examples per concept from the test set. This corresponds to a set of 600 examples. We compute the feature importance for these examples only.

Alternative architecture. We extended the analysis of Section 3.1.1 to a ResNet-50 architecture. We fine-tune the ResNet model on the CUB dataset and reproduced the experiment from Section 3.1.1 with this new architecture. In particular, we fit a CAR and a CAV classifier on the penultimate layer of the ResNet. We then measure the accuracy averaged over the CUB concepts. This results in $(89 \pm 1)\%$ accuracy for CAR classifiers and $(87 \pm 1)\%$ accuracy for CAV classifiers. We deduce that CAR classifiers are highly accurate to identify concepts in the penultimate ResNet layer. As in the main paper, we observe that CAR classifiers outperform CAV classifiers, although the gap is smaller than for the Inception-V3 neural network. We deduce that our CAR formalism extends beyond the architectures explored in the paper and we hope that CAR will become widely used to interpret any more architectures.

F Use Case

This appendix provides useful details to reproduce the use case from Section 3.2.

Computing Resources. The use case was run on a single machine equipped with a 18-Core Intel Core i9-10980XE CPU and a NVIDIA RTX A4000 GPU. The machine runs on Python 3.9 [76] and Pytorch 1.10.2 [77].

Table 5: MNIST Concepts

Concept Class	Loop	Vertical Line	Horizontal Line	Curvature
0	✓	✗	✗	✓
1	✗	✓	✗	✗
2	✓	✗	✗	✓
3	✗	✗	✗	✓
4	✗	✓	✓	✗
5	✗	✗	✓	✓
6	✓	✗	✗	✓
7	✗	✓	✓	✗
8	✓	✗	✗	✓
9	✓	✗	✗	✓

Table 6: SEER Model

Layer Type	Hyperparameters	Activation
Linear	Input Features = 21, Output Features = 400	ReLU
Dropout	p = 0.3	
Linear	Input Features = 400, Output Features = 100	ReLU
Dropout	p = 0.3	
Linear	Input Features = 100, Output Features = 2	

Dataset license. The SEER dataset is made available under the terms of the SEER Research Data Use Agreement.

Model. The detailed architecture of the model is provided in Table 6.

Data split. We randomly split the whole SEER dataset into a training set (90% of the data) and a test set (the remaining 10%). Since patients with a death outcome are in minority (less than 3%), we oversample them to obtain a balanced training set.

Model fitting. In fitting the model, we use the test set as a validation set since our purpose is not to obtain the model with the best generalization but simply a model that performs well on a set of examples we wish to explain (here the examples from the test set). The model is trained to minimize the cross-entropy between its prediction and the true labels. We use a Adam optimizer with batches of 500 examples, a learning rate of 10^{-3}, a weight decay of 10^{-5} for 500 epochs with patience 50.

Concepts. The concepts correspond to prostate cancer grades. Those grades can be computed from the Gleason score as follows [70]:

$$\mathrm{Grade}(\mathrm{Gleason}_1, \mathrm{Gleason}_2) = \begin{cases} 1 & \text{if } \mathrm{Gleason}_1 + \mathrm{Gleason}_2 \leq 6 \\ 2 & \text{if } \mathrm{Gleason}_1 = 3 \ \wedge \ \mathrm{Gleason}_2 = 4 \\ 3 & \text{if } \mathrm{Gleason}_1 = 4 \ \wedge \ \mathrm{Gleason}_2 = 3 \\ 4 & \text{if } \mathrm{Gleason}_1 + \mathrm{Gleason}_2 = 8 \\ 5 & \text{if } \mathrm{Gleason}_1 + \mathrm{Gleason}_2 \geq 9. \end{cases}$$

It goes without saying that the model is trained with the Gleason scores only, not with the grades.

Concept classifiers. We fit a SVC with linear kernel and default hyperparameters from scikit-learn.

Concept-based feature importance. We use Captum's implementation [72] of Integrated Gradients [25] with default parameters.

G Explanation Robustness

Adversarial perturbations. We perform an experiment to evaluate the robustness of CAR explanations with respect to adversarial perturbations. In this experiment, we work with the MNIST dataset in the same setting as the experiment from Section 3.1.2. We train a CAR concept classifier for each MNIST concept $c \in [C]$. We use the CAR classifier to output TCAR scores relating the concept c with each class $k \in [d_Y]$. As in the main paper, since the ground-truth association between concepts and classes is known (e.g. the class corresponding to digit 8 will always have the concept loop), we can compute the correlation $r(\mathrm{TCAR}, \mathrm{TrueProp})$ between our TCAR score and the ground-truth proportion of examples that exhibit the concept. In this experiment, the correlation is evaluated on a test set $\mathcal{D}_{\text{test}} = \mathcal{D}_{\text{adv}} \sqcup \mathcal{D}_{\text{orig}}$ that contains adversarial test examples $\mathcal{D}_{\text{adv}}$ and original test examples $\mathcal{D}_{\text{orig}}$. Each adversarial MNIST image $\boldsymbol{x}_{\text{adv}} \in \mathcal{D}_{\text{adv}}$ is constructed by finding a small (w.r.t. the $\| \cdot \|_{\infty}$ norm) perturbation $\boldsymbol{\epsilon} \in \mathbb{R}^{d_X}$ around an original test image $\boldsymbol{x} \in \mathcal{X}$ that maximizes the prediction shift for the model $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$:

$$\boldsymbol{\epsilon} = \arg \max_{\tilde{\boldsymbol{\epsilon}} \in \mathbb{R}^{d_X}} \mathrm{CrossEntropy}[\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x} + \tilde{\boldsymbol{\epsilon}})] \ s.t. \ \|\tilde{\boldsymbol{\epsilon}}\|_{\infty} < .1$$

Table 7: MNIST Adversarial Perturbation Sensitivity

Adversarial %	$r(\mathrm{TCAR}, \mathrm{TrueProp})$
0	.99
5	.99
10	.99
20	.99
50	.97
70	.95
100	.90

The adversarial image is then defined as $\boldsymbol{x}_{\mathrm{adv}} \equiv \boldsymbol{x} + \boldsymbol{\epsilon}$. We measure the correlation $r(\mathrm{TCAR}, \mathrm{TrueProp})$ by varying the proportion $\frac{|\mathcal{D}_{\mathrm{adv}}|}{|\mathcal{D}_{\mathrm{test}}|}$ of adversarial examples in the test set. The results are reported in Table 7.

We observe that the TCAR scores keep a high correlation with the true proportion of examples that exhibit the concept even when all the test examples are adversarially perturbed. We conclude that TCAR explanations are robust with respect to adversarial perturbations in this setting.

Background shift. For completeness, we have also adapted the background shift robustness experiment in Section 7 from [27]. We use CAR to explain the predictions of our Inception-V3 model trained on the original CUB training set. The explanations are made on test images where the background has been replaced. As [27], we use the segmentation of the CUB dataset to isolate the bird on each image. The rest of the image is replaced by a random background sampled from the *Place365* dataset [80]. This results in a test set $\mathcal{D}_{\mathrm{test}}$ with a background shift with respect to the training set. By following the approach from Section 3.1.2 of our paper, we measure the correlation $r(\mathrm{TCAR}, \mathrm{TrueProp})$ between the TCAR score and the true proportion of examples in the class that exhibit the concept for each $(\mathrm{class}, \mathrm{concept})$ pair. We measured a correlation of $r(\mathrm{TCAR}, \mathrm{TrueProp}) = .82$ in the background-shifted test set. This is close to the correlation for the original test set reported in the main paper, which suggests that CAR explanations are robust with respect to background shifts. Note that this correlation is still better than the one obtained with TCAV on the original test set.

H Using CAR to Understand Unsupervised Concepts

Our CAR formalism adapts to a wide variety of neural network architectures. In this appendix, we use CAR to analyze the concepts discovered by a self explaining neural network (SENN) trained on the MNIST dataset. As in [81], we use a SENN of the form

$$f(\boldsymbol{x}) = \sum_{s=1}^{S} \theta_s(\boldsymbol{x}) \cdot g_s(\boldsymbol{x}),$$

Where $g_s(\boldsymbol{x})$ and $\theta_s(\boldsymbol{x})$ are respectively the activation and the relevance of the synthetic concept $s \in [S]$ discovered by the SENN model. We follow the same training process as [81]. This yields a set of $S = 5$ concepts explaining the predictions made by the SENN $f : \mathcal{X} \to \mathcal{Y}$.

We use our CAR formalism to study how the synthetic concepts $s \in [S]$ discovered by the SENN are related to the concepts $c \in \{\mathrm{Loop}, \mathrm{Vertical\ Line}, \mathrm{Horizontal\ Line}, \mathrm{Curvature}\}$ introduced in our paper. With our formalism, the relevance of a concept c for a given prediction $\boldsymbol{x} \mapsto f(\boldsymbol{x})$ is measured by the concept density $\rho^c \circ \boldsymbol{g}(\boldsymbol{x})$. To analyze the relationship between the SENN concept s and the concept c, we can therefore compute the correlation of their relevance:

$$r(s, c) = \mathrm{corr}_{\mathbf{X} \sim P_{\mathrm{empirical}}(\mathcal{D}_{\mathrm{test}})}[\theta_s(\mathbf{X}), \rho^c \circ \boldsymbol{g}(\mathbf{X})].$$

When this correlation increases, the concepts s and c tend to be relevant together more often. We report the correlation between each pair (s, c) in Table 8.

We note the following:

Table 8: SENN Concepts Correspondence

Correlation $\mathrm{r}(\mathrm{s}, \mathrm{c})$	Loop	Vertical Line	Horizontal Line	Curvature
SENN Concept 1	-0.28	-0.12	0.26	0.11
SENN Concept 2	-0.50	0.71	-0.03	-0.69
SENN Concept 3	-0.47	0.10	0.71	-0.14
SENN Concept 4	-0.33	0.02	-0.06	-0.01
SENN Concept 5	0.57	-0.00	-0.63	0.07

1. SENN Concept 2 correlates well with the Vertical Line Concept.
2. SENN Concept 3 correlates well with the Horizontal Line Concept
3. SENN Concept 5 correlates well with the Loop Concept.
4. SENN Concepts 1 and 4 are not well covered by our concepts.

The above analysis shows the potential of our CAR explanations to better understand the abstract concepts discovered by SENN models. We believe that the community would greatly benefit from the ability to perform similar analyses for other interpretable architectures, such as disentangled VAEs.

I Using CAR with NLP

CAR is a general framework and can be used in a wide variety of domains that involve neural networks. In the main paper, we show that CAR provides explanations for various modalities:

1. Large image dataset
2. Medical time series
3. Medical tabular data

We now perform a toy experiment to assess if those conclusions extend to the NLP setting. We train a small CNN on the IMDB Review dataset to predict whether a review is positive or negative. We use GloVe [82] to turn the word tokens into embeddings. We would like to assess whether the concept $c = \text{Positive Adjective}$ is encoded in the model's representations. Examples that exhibit the concept c are sentences containing positive adjectives. We collect a positive set $\mathcal{P}^c$ of $N^c = 90$ such sentences. The negative set $\mathcal{N}^c$ is made of N^c sentences randomly sampled from the Gutenberg Poem Dataset. We verified that the sentences from $\mathcal{N}^c$ did not contain positive adjectives. We then fit a CAR classifier on the representations obtained in the penultimate layer of the CNN.

We assess the generalization performance of the CAR classifier on a holdout concept set made of $N^c = 30$ concept positive and negative sentences (60 sentences in total). The CAR classifier has an accuracy of 87% on this holdout dataset. This suggests that the concept c is smoothly encoded in the model's representation space, which is consistent with the importance of positive adjectives to identify positive reviews. We deduce that our CAR formalism can be used in a NLP setting. We believe that using CARs to analyze large-scale language model would be an interesting study that we leave for future work.

J Increasing Explainability at Training Time

Improving neural networks explainability at training time constitutes a very interesting area of research but is beyond the scope of our paper. That said, we believe that our paper indeed contains insights that might be the seed of future developments in neural network training. As an illustration, we consider an important insight from our paper: the fact that the accuracy of concept classifiers seems to increase with the depth of the layer for which we fit a classifier. In the main paper, this is mainly reflected in Figure 4. This observation has a crucial consequence: it is not possible to reliably characterize the shallow layers in terms of the concepts we use.

In order to improve the explainability of those shallow layers, one could leverage the recent developments in contrastive learning. The purpose of this approach would be to separate the concept set

$\mathcal{P}^c$ and $\mathcal{N}^c$ in the representation space $\mathcal{H}$ corresponding to a shallow layer of the neural network. A practical way to implement this would be to follow [79]. Assume that we want to separate concept positives and negatives in the representation space $\mathcal{H}$ induced by the shallow feature extractor $\boldsymbol{g} : \mathcal{X} \to \mathcal{H}$. As [79], one can use a projection head $\mathbf{p} : \mathcal{H} \to \mathcal{Z}$ and enforce the separation of the concept sets through the contrastive loss

$$\mathcal{L}^c_{\text{cont}} = \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in (\mathcal{P}^c)^2} - \log \frac{\exp(\tau^{-1} \cdot \cos[\mathbf{p} \circ \boldsymbol{g}(\boldsymbol{x}_i), \mathbf{p} \circ \boldsymbol{g}(\boldsymbol{x}_j)])}{\sum_{\boldsymbol{x}_k \in (\mathcal{P}^c \cup \mathcal{N}^c) \setminus \{\boldsymbol{x}_i\}} \exp(\tau^{-1} \cdot \cos[\mathbf{p} \circ \boldsymbol{g}(\boldsymbol{x}_i), \mathbf{p} \circ \boldsymbol{g}(\boldsymbol{x}_k)])}, \tag{6}$$

where $\cos(\mathbf{z}_1, \mathbf{z}_2) \equiv \frac{\mathbf{z}_1^\intercal \mathbf{z}_2}{\|\mathbf{z}_1\|_2 \cdot \|\mathbf{z}_2\|_2}$ and $\tau \in \mathbb{R}^+$ is a temperature parameter. The effect of this loss is to group the concept positive examples from $\mathcal{P}^c$ together and apart from the concept negatives $\mathcal{N}^c$ in the representation space $\mathcal{H}$. To the best of our knowledge, concept-based contrastive learning has not been explored in the literature. We believe that it would constitute an interesting contribution to the field based on the insights from our paper.

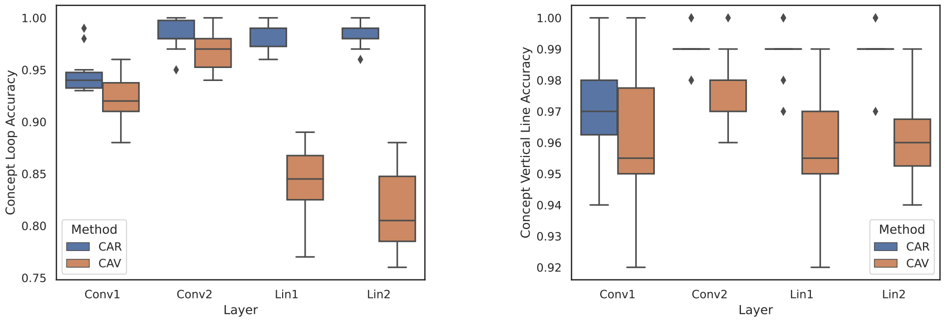
K Further Examples

In this appendix, we provide several examples to illustrate the experiments from Section 3.

Concept classifiers. The accuracy of the concept classifiers for all the MNIST and ECG concepts are given in Figures 11 and 12. Each box-plot is built with 10 random seeds where the concept sets $\mathcal{P}^c$ and $\mathcal{N}^c$ are allowed to vary. We observe that the CAR classifiers are more accurate for each of the observed concepts

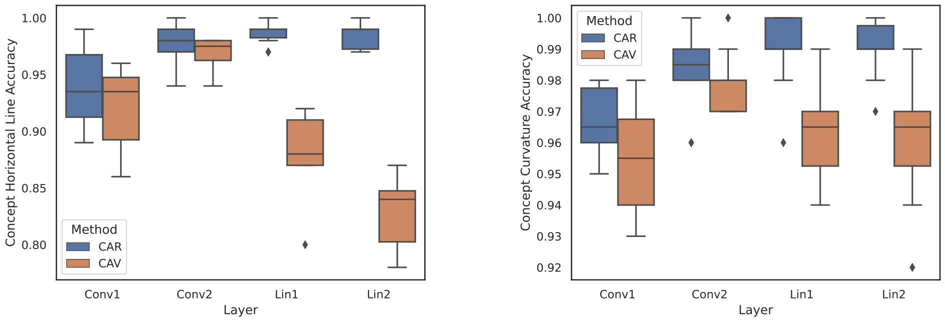
Global explanations. The global concept explanations for all the MNIST concepts and some of the CUB concepts are given in Figures 13 and 14. As the examples from Section 3.1.2, we see that TCAR explanations are more consistent with the human concept annotations.

Saliency maps. Examples of concept-based an vanilla saliency maps for MNIST, ECG and CUB examples are given in Figures 15 , 16 , 17 , 18 and 19. As explained in the main paper, we observe that concept-based saliency maps are indeed distinct form vanilla saliency maps. Furthermore, saliency maps for different concepts are not interchangeable. In the CUB case, we note that concept are not always identified with the minimal amount of features (e.g. some of the saliency maps from Figure 19 highlight pixels that do not always belong to the bird's breast). This surprising observation seems to occur even for concept-bottleneck models that are explicitly trained to recognize concepts [44]. We believe that this could be improved by training concept classifiers with images that include a segmentation highlighting the concept of interest (e.g. the bird's breast). We leave this idea for future works.



(a) Loop concept

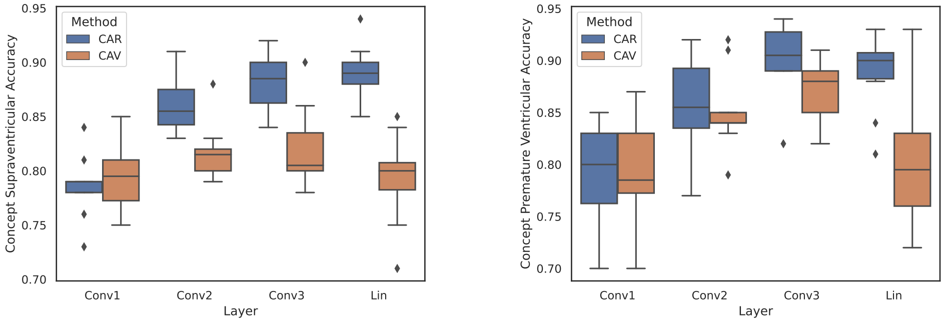
(b) Vertical line concept



(c) Horizontal line concept

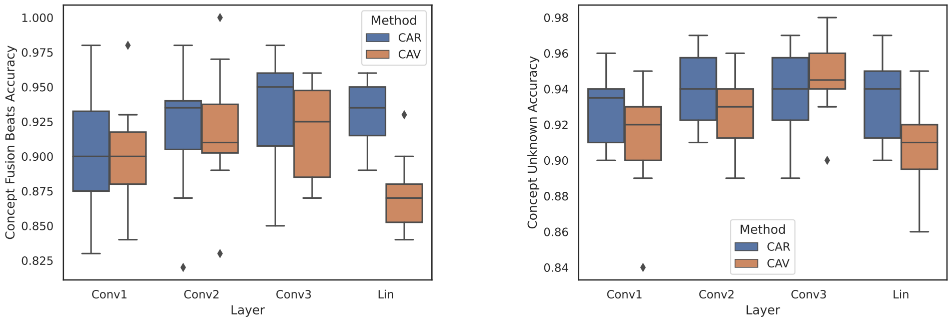
(d) Curvature concept

Figure 11: Concept accuracy for MNIST concepts



(a) Supraventricular concept

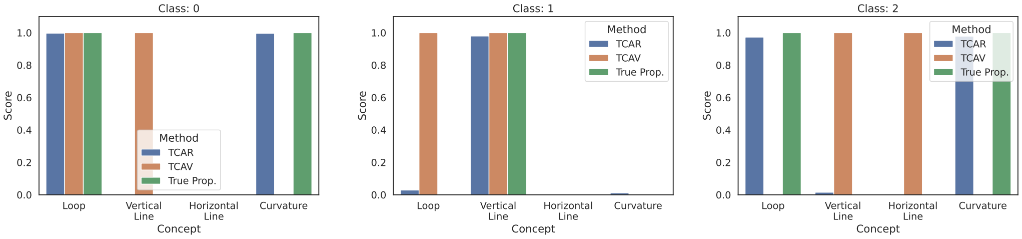
(b) Premature ventricular concept



(c) Fusion beats concept

(d) Unknown beat concept

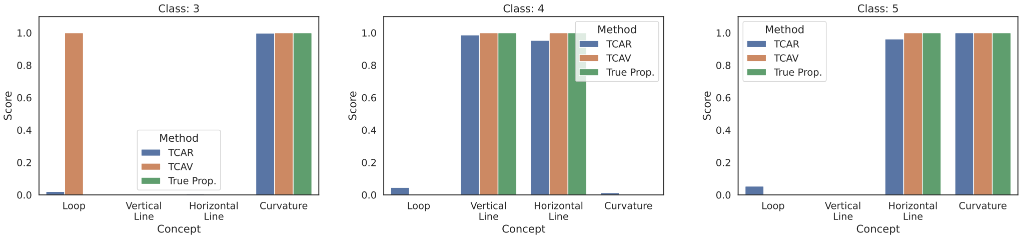
Figure 12: Concept accuracy for ECG concepts



(a) Class 0

(b) Class 1

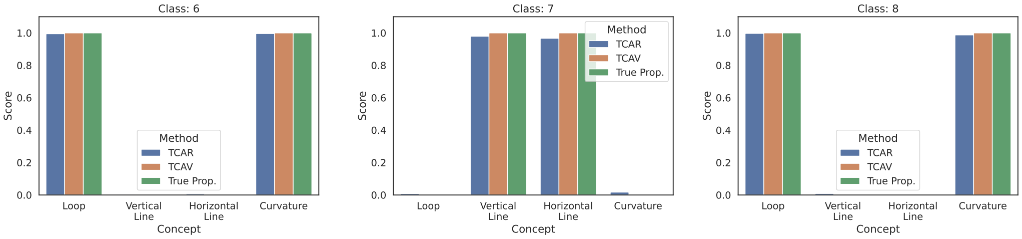
(c) Class 2



(d) Class 3

(e) Class 4

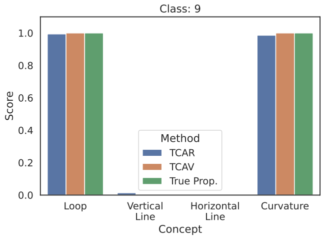
(f) Class 5



(g) Class 6

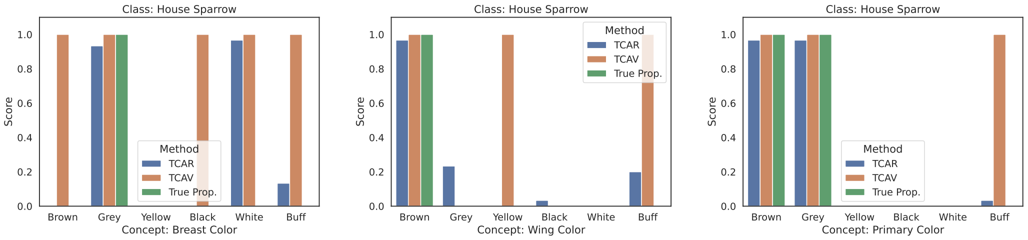
(h) Class 7

(i) Class 8

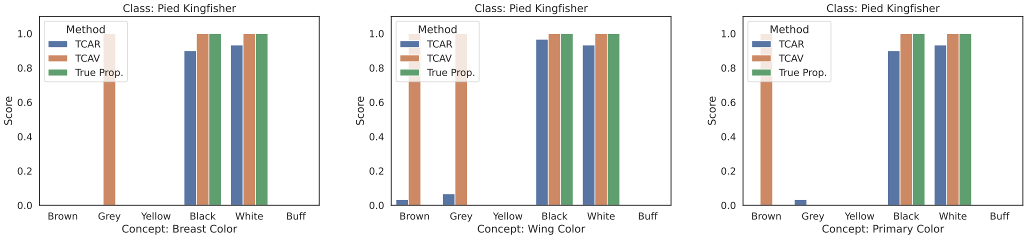


(j) Class 9

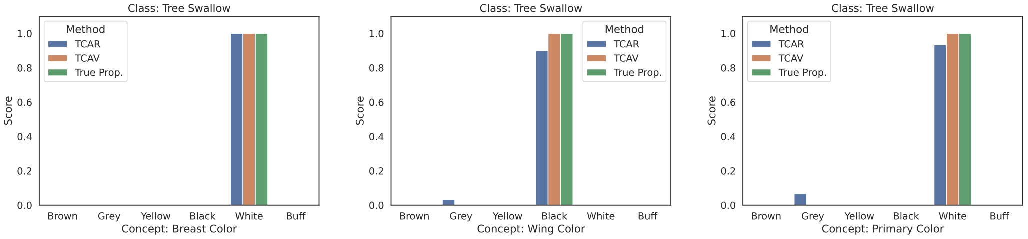
Figure 13: Global concept explanations for MNIST



(a) House sparrow breast colour (b) House sparrow wing colour (c) House sparrow primary colour

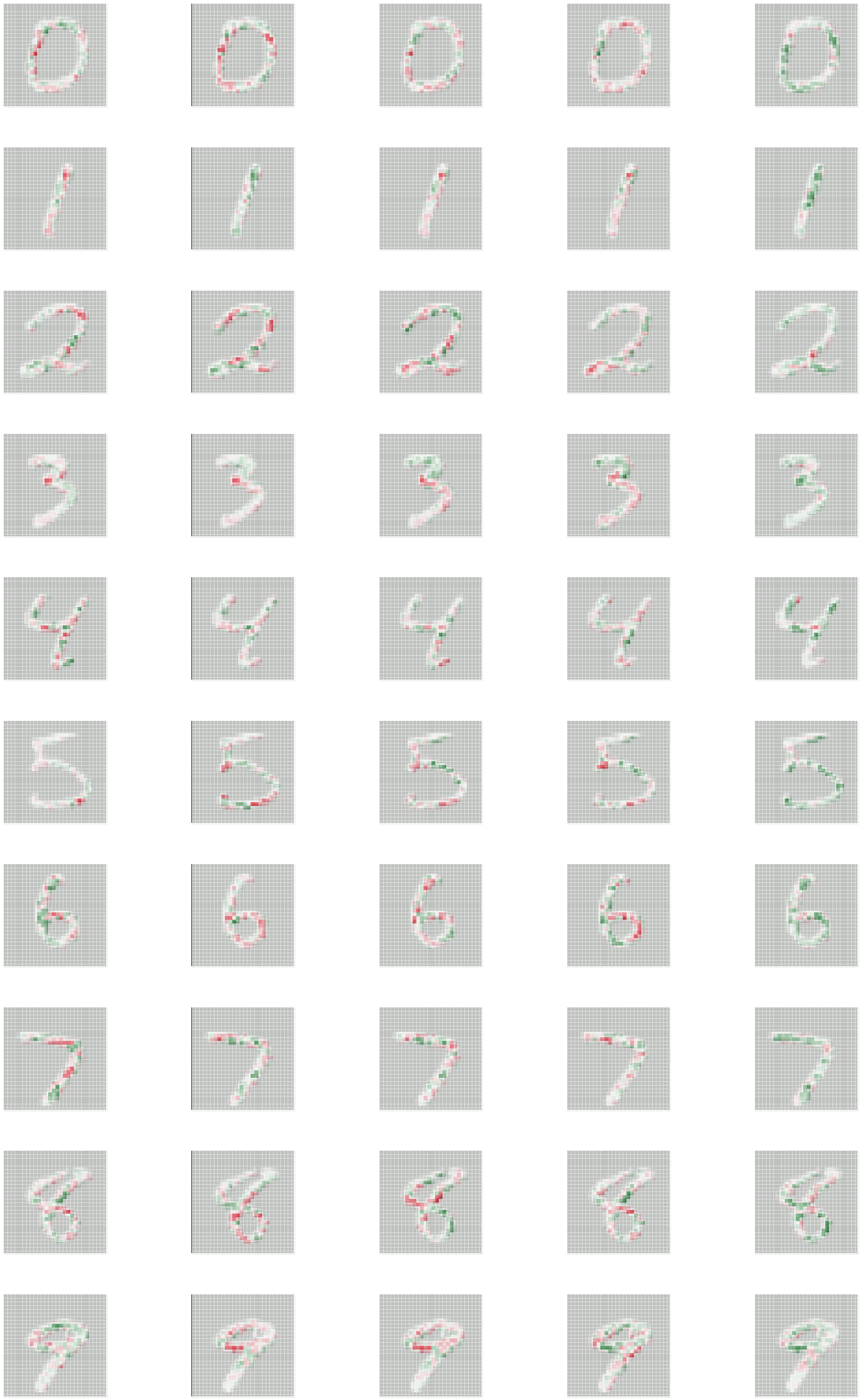


(d) Pied kingfisher breast colour (e) Pied kingfisher wing colour (f) Pied kingfisher primary colour



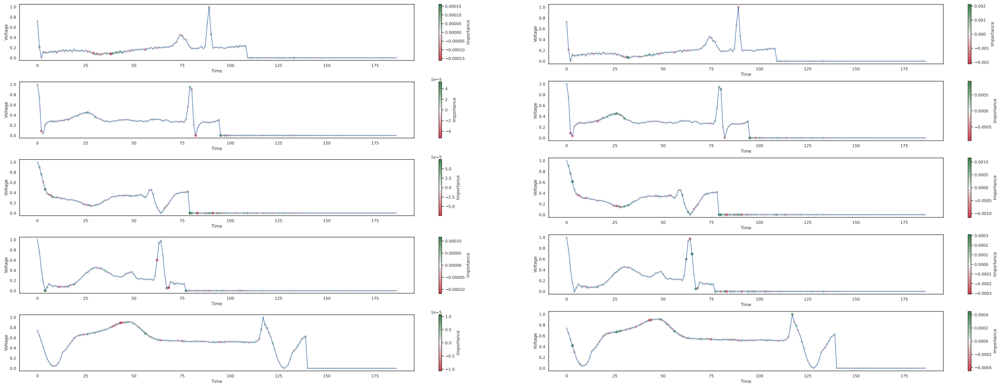
(g) Tree swallow breast colour (h) Tree swallow wing colour (i) Tree swallow primary colour

Figure 14: Global concept explanations for CUB

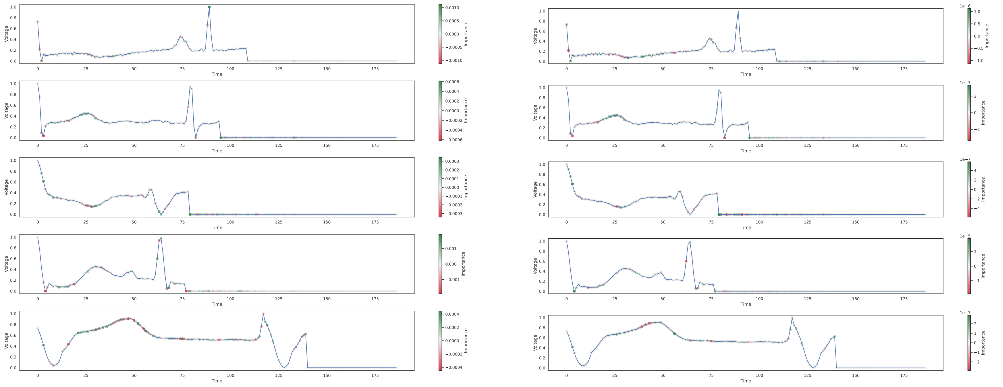


(a) Loop (b) Vertical line (c) Horiz. line (d) Curvature (e) Vanilla

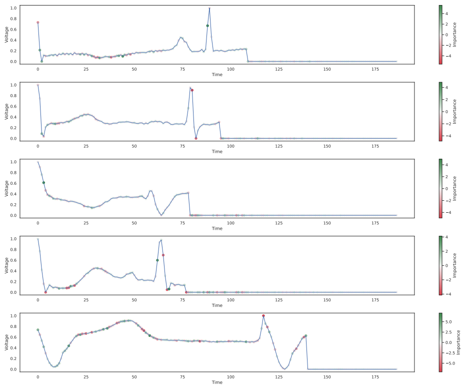
Figure 15: Concept-based saliency maps for MNIST



(a) Supraventricular (b) Premature ventricular

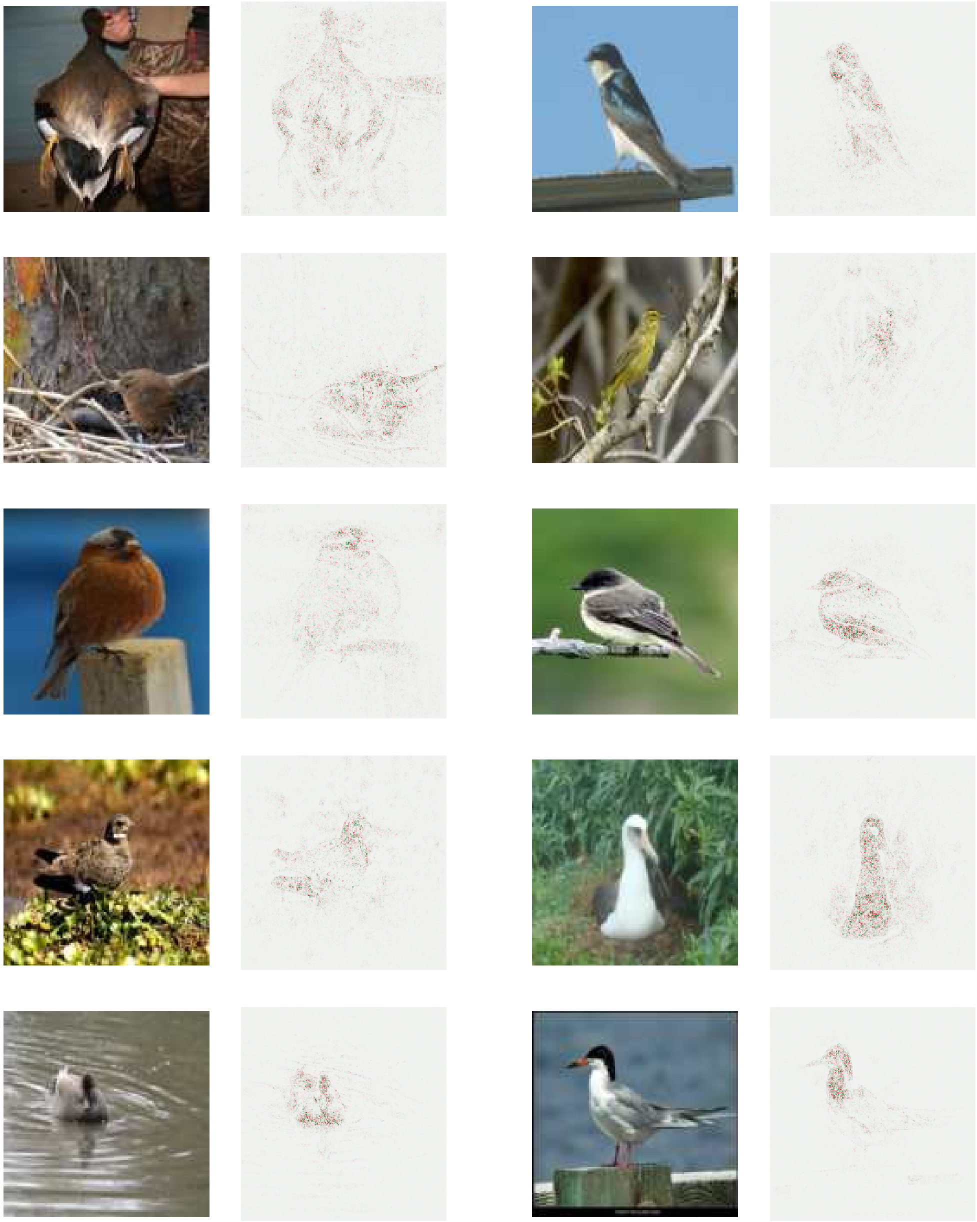


(c) Fusion beats (d) Unknown



(e) Vanilla

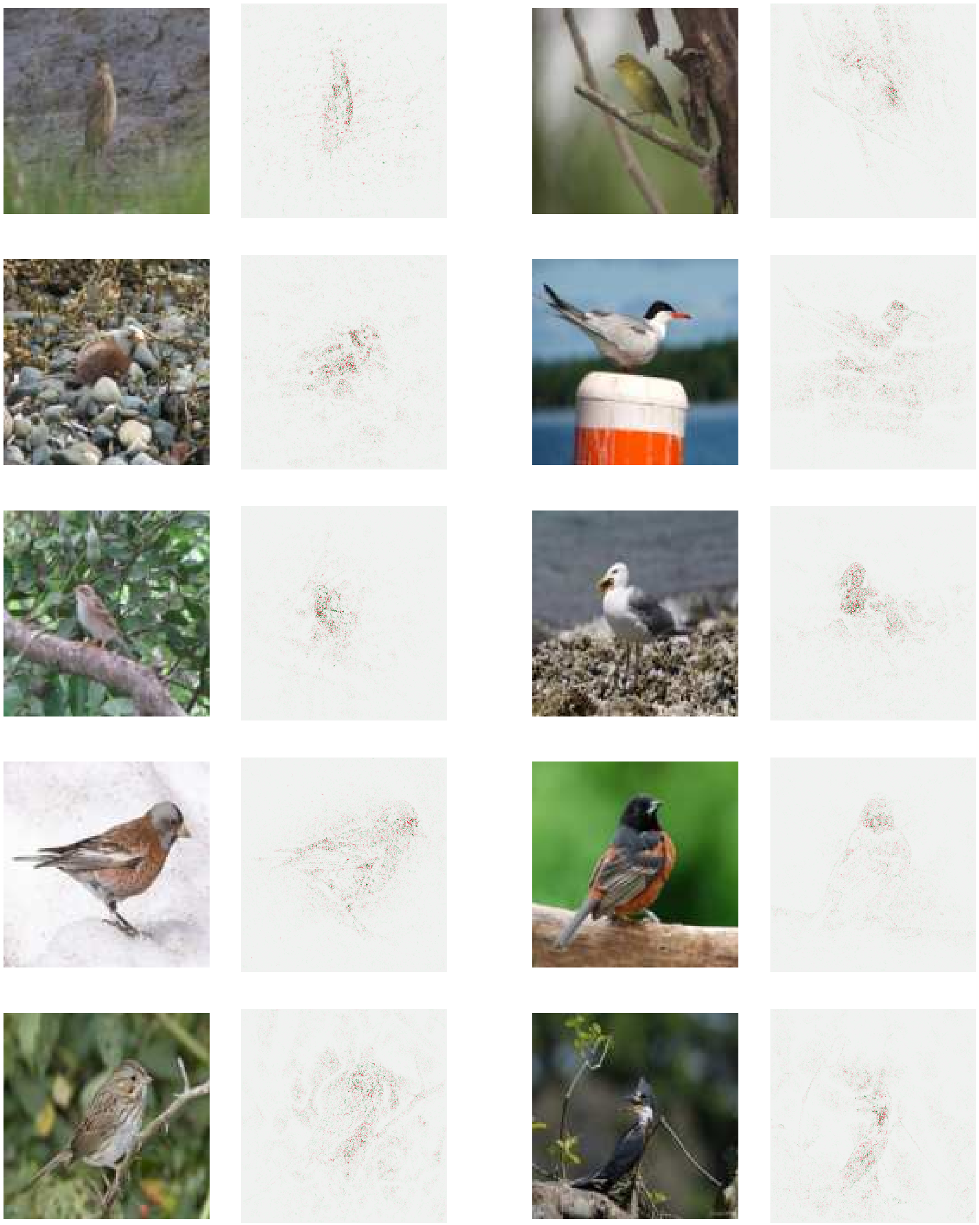
Figure 16: Concept-based saliency maps for ECG



(a) Brown belly

(b) Solid belly pattern

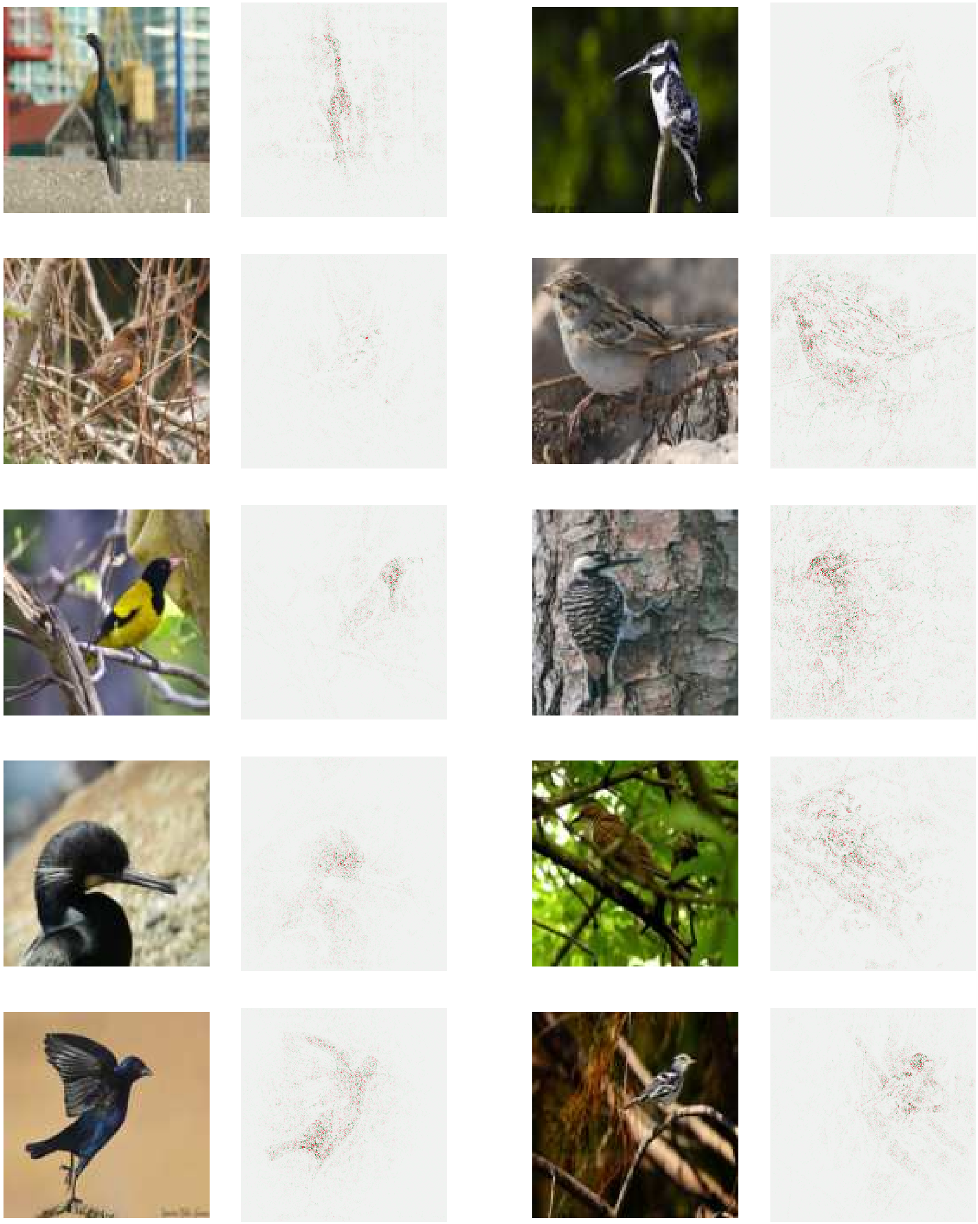
Figure 17: Concept-based saliency maps for CUB (1/3)



(a) Stripped back pattern

(b) Solid back pattern

Figure 18: Concept-based saliency maps for CUB (2/3)



(a) Black breast

(b) White breast

Figure 19: Concept-based saliency maps for CUB (3/3)