Supplement: Efficient Data Source Relevance Quantification for Multi-Source Neural Networks

Jakob Gawlikowski[1,2]
jakob.gawlikowski@dlr.de
Nina Maria Gottschling[3]
nina-maria.gottschling@dlr.de

[1] Technical University of Munich
Data Science in Earth Observation
Munich, Germany

[2] German Aerospace Center (DLR)
Institute of Data Science
Jena, Germany

[3] German Aerospace Center (DLR)
Remote Sensing Technology Institute
Oberpfaffenhofen, Germany

Contents

1 Relevance Forward Propagation

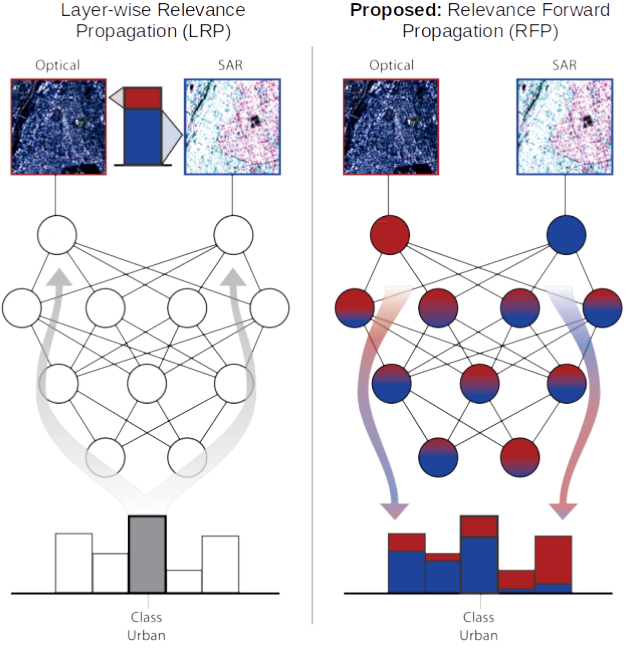

Figure S1: **Left**: Layer-wise Relevance Propagation (LRP) propagates the relevance of an individual output backwards through the network onto the input data, where it is source-wise aggregated. **Right (proposed)**: Relevance Forward Propagation (RFP) approach. The relevance representation of the input and the online linearization propagates the data source through the network. The output is given as predictions decomposed into the individual data source relevance values.

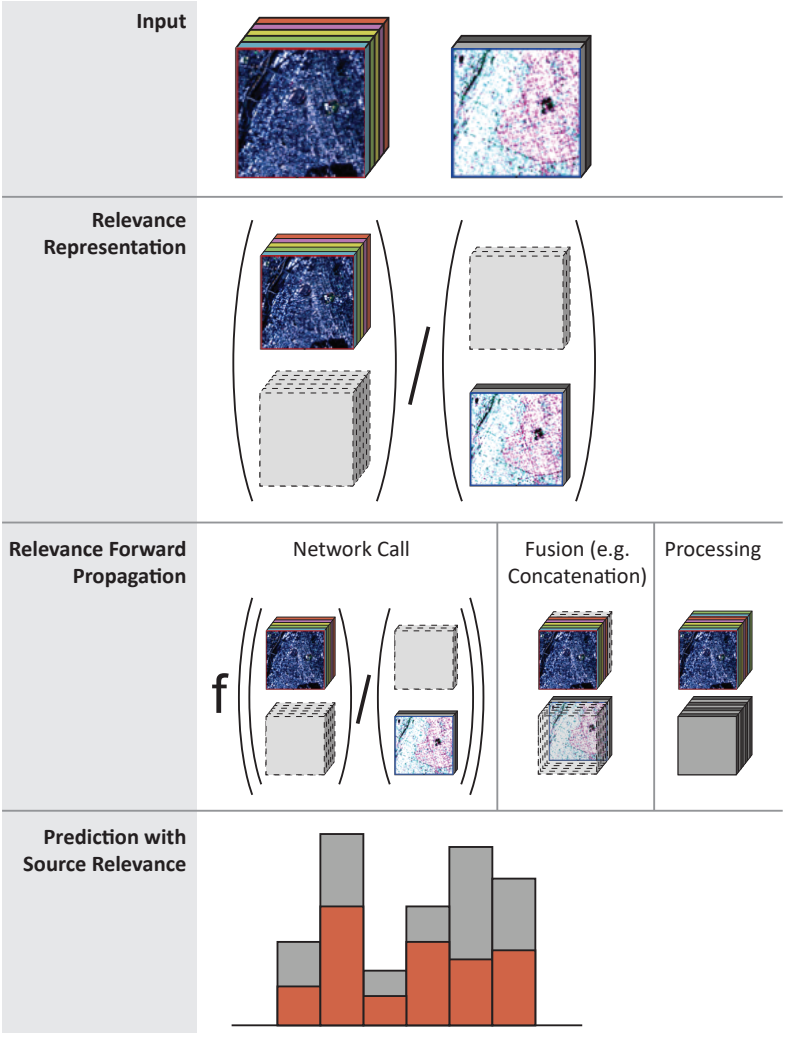

Figure S2: This method sketches the main steps of the Relevance Forward Propagation approach. The input data is mapped into relevance representation and the data source relevance is propagated through the network onto the model output.

2 Summary of other relevant approaches

2.1 Perceptual Score

Gat et al. introduce perceptual score [5]. The perceptual score for a modality M_m, or also data source, of a sample value-target pair (x, y) is defined as the change in accuracy caused by modality M_m,

$$P_{f,x,y}(M_m) = \mathrm{Acc}_{f,x,y}(\mathcal{M}) - \mathrm{Acc}_{f,x,y}(\mathcal{M} \setminus \{M_m\}) .$$

The accuracy with the left-out modality is here approximated by the expected accuracy when choosing the input for this modality randomly from the dataset, i.e.,

$$\mathbb{E}_{(x,y)}\left[\mathrm{Acc}_{f,x,y}\left(\mathcal{M}\setminus\{M_m\}\right)\right].$$

The perceptual score of a model is defined as the normalized expectation of the sample

$$P_{f,\mathcal{D}}(M_m) = \frac{1}{Z}\left(\mathbb{E}_{(x,y)}\left[P_{f,x,y}\left(M_m\right)\right]\right) .$$

Consistency of the Perceptual Score Based on experiments, we set the number of samples to approximate the perceptual score to 200. Fig. S3 shows the deviation for the individual classes of the Basic Multi-Source MNIST dataset. For SEN12MS, the evaluation looked similar; the ADVANCE dataset is a special case, as the relevance is unbalanced.

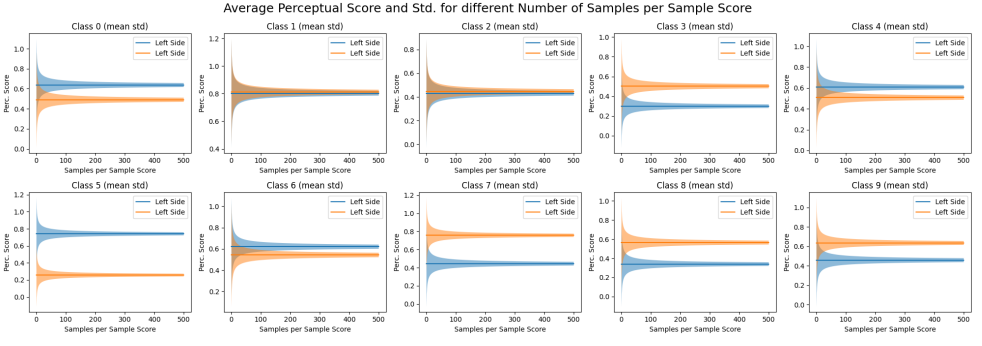

Figure S3: Perceptual Score and expected standard deviation of the resulting score based on the number of samples per sample score.

2.2 SHAPE

Hu et al. [8] introduce a similar approach motivated by game theory with the extension that shape can quantify the cooperation among multiple sources by computing the score by replacing individual and groups of data sources with zero input values and then do a complete dataset performance evaluation based on the modified data. The score for a data source or a group of data sources is the average performance without replacing the input data with zeros minus the performance when the data is replaced by zeros. This leads to improved performance, as no sampling of other data source samples is needed and allows the consideration of subsets of data sources. On the downside, this is only applicable on whole datasets and not on individual examples.

2.3 LRP and its extensions

The basic idea of Layer-wise Relevance Propagation (LRP) [1] is to layer-wise back-propagate the neural network output $f(x)$, which is decomposed into relevance values, onto the input data. The relevance values are subject to a conservation property, i.e., the relevance R_j of a neuron j at layer $l+1$ has to be fully distributed over all neurons k (including the bias)

of layer l that are connected to neuron j. We indicate this connection by a weight ω_{kj} scaling the activation a_k from neuron k, which is passed to neuron j. We define the set of all outgoing connections by $C^+(k) = \{l \mid \exists\ \omega_{kl}\}$ and the set of all incoming connections by $C^-(j) = \{l \mid \exists\ w_{lj}\}$. The relevance back-propagation is given by

$$R_k^l = \sum_{j \in C^+(k)} \frac{a_k \left(\omega_{kj} + \gamma \omega_{kj}^+\right)}{\varepsilon + \sum\limits_{\tilde{l} \in C^-(k)} a_l \left(\omega_{lj} + \gamma \omega_{lj}^+\right)} R_j^{l+1} ,$$

for some $\varepsilon, \gamma \geq 0$ and $^+$ being the clipping to the non-negative space. The input relevance values R_j^0 are obtained by back-propagating the relevance values of the input layer $l = 1$ to all input points j. Based on this, extensions of LRP that reduce noise and enhance positive contributions are summarized in [10], namely *LRP-0* ($\varepsilon = 0, \gamma = 0$), *LRP-*$\varepsilon$ ($\gamma = 0$) and *LRP-*γ ($\gamma \neq 0$). For LRP and its variants and for n data sources, the corresponding source-wise relevance values $R^{0,1}, \ldots, R^{0,n}$ at $l = 0$ can be determined by summing up the individual data relevance values of the input points of the corresponding data source, i.e., $R^{0,s} = \sum_j R_j^{0,s}$ for $s = 1, \ldots, n$.

2.4 Integrated Gradients

Integrated Gradients attributes input relevance by integrating the gradients of the model's output with respect to its input over a line from a baseline input to the actual input. The baseline is usually chosen to only contain zero values. For an input x and a baseline x_0, the integral is approximated by a sum of $n+1$ samples, defined as

$$x_i = x_0 + \frac{1}{n} \cdot (x - x_0) .$$

The integrated gradients are then computed as the average gradient, weighted by the difference to the baseline, i.e., for a neural network f and $\circ$ as the point-wise multiplication, we get

$$\text{Integrated Gradients} = (x - x_0) \circ \frac{1}{n+1} \sum_{i=0}^{n} \frac{\partial f(x_i)}{\partial x_i} .$$

3 Proofs and Examples

3.1 Local Linearization for RFP

Definition 1. *[Input Relevance Values]*
For an input sample $x = (x^1, ..., x^n) \in \mathcal{S}$ we define the input relevance representation as

$$r_{l=0}(x) := \tilde{x} = \begin{pmatrix} \tilde{x}^1 \\ \cdots \\ \tilde{x}^n \end{pmatrix} = \begin{pmatrix} [0_{\mathcal{S}_1},\ x^1,\ 0_{\mathcal{S}_1},\ ...] \\ ... \\ [0_{\mathcal{S}_n},\ ...,\ 0_{\mathcal{S}_n},\ x^n] \end{pmatrix} , \tag{1}$$

where $0_{\mathcal{S}_i}$ are zero tensors with the same shape as the data samples of the source $\mathcal{S}_i$ and $[\cdot]$ is the stacking operator. It holds that $\sum_{i=0}^{n} \tilde{x}_i^s = x^s$ and the individual terms of the sum represent the relevance of the corresponding data sources.

Definition 2 (Propagated Relevance Values and Local Linearization)**.**
Fix $x \in \mathcal{S}$, $k \in \{1, ..., L\}$ and a k-th operation $\ell_k : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ in a neural network. The layer-wise forward propagated relevance values are defined (recursively) as

$$r^s_{l=k+1}(x) := \tilde{\ell}^x_k(r^s_{l=k}(x)) \qquad \text{for } k \geq 0 , \tag{2}$$

for each $s \in \{0, ..., n\}$ and where $\tilde{\ell}^x_k : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ is called a local linearization of ℓ_k that satisfies

$$\ell_k\left(\sum_{s=0}^{n} r^s_{l=k}(x)\right) = \tilde{\ell}^x_k\left(\sum_{s=0}^{n} r^s_{l=k}(x)\right) = \sum_{s=0}^{n} \tilde{\ell}^x_k(r^s_{l=k}(x)) , \tag{3}$$

for fixed $x \in \mathcal{S}$. The recursion basis $r^s_{l=0}(x)$ is the input relevance as defined in (1).

Note that the local linearization according to (3) is not unique. For piece-wise linear layers and activation functions, the linearization is straightforward. However, linearization schemes must be chosen for all other activation functions. In the following, we show approaches that utilize first-order Taylor approximation and a weighting of the inputs. It is important to note that both approaches result in the same approximation for piece-wise linear functions that are only non-differentiable in zero. The weighted linearization scheme directly follows the principles of LRP, such that the properties of Proposition 2 still hold.

3.1.1 Linearization with Taylor Approximation

If a network architecture has a differentiable activation function ρ, a possible local linearization is given in the following. This linearization assigns the data source relevance to the individual data sources and then attributes the difference as bias. It is also possible to separate the difference based on the input relevance values of the individual data sources - the adjustment for this is straightforward.

The linearization with first-order Taylor approximation is defined as follows:
Let ρ_k, $k \in \{1, ..., L\}$ be an operator in a neural network and let ρ_k be differentiable for a

network input $x \in \mathcal{S}$. Then, the local linearization $\tilde{l}_k^x$ satisfying (3) is obtained by

$$\tilde{\ell}_k^x(y,s) = \begin{cases} \rho'\left(\sum_{s=0}^n r_{l=k}^s(x)\right) r_{l=k}^s(x) & s \in \{1,\ldots,n\}, \quad y = r_{l=k}^s(x) \\ \rho\left(\sum_{s=0}^n r_{l=k}^s(x)\right) - \rho'\left(\sum_{s=0}^n r_{l=k}^s(x)\right) \sum_{s=1}^n r_{l=k}^s(x) & s = 0, \quad y = r_{l=k}^0(x) \\ \rho\left(\sum_{s=0}^n r_{l=k}^s(x)\right) & s = \emptyset, \quad y = \sum_{s=0}^n r_{l=k}^s(x) \end{cases} \tag{4}$$

where ρ' is the derivative function of ρ.

Proof. The left equation of eq. (3) is satisfied by the local linearization in (5), as for $y = \sum_{s=0}^n r_{l=k}^s(x)$ we have that

$$\ell_k\left(\sum_{s=0}^n r_{l=k}^s(x)\right) = \rho\left(\sum_{s=0}^n r_{l=k}^s(x)\right) = \tilde{\ell}_k^x\left(\sum_{s=0}^n r_{l=k}^s(x), s=\emptyset\right).$$

The right equation of eq. (3) is satisfied by the local linearization in (5), as we have that

$$\begin{aligned} \sum_{s=0}^n \tilde{\ell}_k^x(r_{l=k}^s(x),s) &= \ell_k^x\left(r_{l=k}^0(x),0\right) + \sum_{s=1}^n \tilde{\ell}_k^x(r_{l=k}^s(x),s) \\ &= \rho\left(\sum_{s=0}^n r_{l=k}^s(x)\right) - \rho'\left(\sum_{s=0}^n r_{l=k}^s(x)\right) \sum_{s=1}^n r_{l=k}^s(x) + \sum_{s=1}^n \rho'\left(\sum_{s=0}^n r_{l=k}^s(x)\right) r_{l=k}^s(x) \\ &= \rho\left(\sum_{s=0}^n r_{l=k}^s(x)\right). \end{aligned}$$

□

Note that the local linearization from Definition 2 is not unique. For example, in (5), one could add some $\varepsilon > 0$ to the derivate of the activation function to circumvent vanishing gradients and enhance numerical stability.

3.1.2 Relevance Weighted Liearization

The input-weighted linearization is the forward equivalent to the LRP rules and is defined as follows. Let ρ an operator at layer l_k, for $k \in \{1,\ldots,L\}$. Fix an input $x \in \mathcal{S}$. Then, the input weighted linearization satisfying (3) is obtained by

$$\tilde{\ell}_k^x(y,s) = \begin{cases} \frac{r_{l=k}^s(x)}{y} \cdot \rho(y) & s \in \{0,1,\ldots,n\},\ y = \sum_{s=0}^n r_{l=k}^s(x) \\ \rho(y) & s = \emptyset,\ y = \sum_{s=0}^n r_{l=k}^s(x)\ . \end{cases} \tag{5}$$

For numerical stability and where needed, we set, when $|y| < \varepsilon$, the relevance of the data sources $s = 1,..,n$ to zero and set the bias relevance to $\rho(y)$.
Further, for activation functions that can become negative (e.g., elu or tanh), one can consider a shift to the non-negative target space (e.g., +1) and attribute this shift to the bias source (-1 for bias relevance). The same can be done the other way around, as the sigmoid function, for example, does not pass the origin.

Proof. For $s = \emptyset$ obvious. For $s \neq \emptyset$ and $|y| > \varepsilon$ we get

$$\sum_{s=0}^{n} \tilde{\ell}_k(y,s) = \sum_{s=0}^{n} \frac{r^s_{l=k}(x)}{y} \cdot \rho(y) = \rho(y)$$

□

3.2 Proof to Lemma 1

Lemma 1 (Piece-Wise Linear Relevance Propagation)**.**
Let $r_{l=k}(x) \in \mathbb{R}^{(n+1)\times d_{in}}$ be a given RFP relevance value in a neural network for $x \in \mathcal{S}$ and $k \in \{1,\ldots,L\}$. Then, for each piece-wise linear operation $l : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ the local linearization is given by the linear operation l in the interval that covers $f_{l=k}(x) = \sum_{s=1}^{n} r^s_{l=k}(x)$.

Proof. The proof follows directly from the definition of piece-wise linearity. □

3.3 Piece-wise Linear Layers Relevance Propagation (Examples)

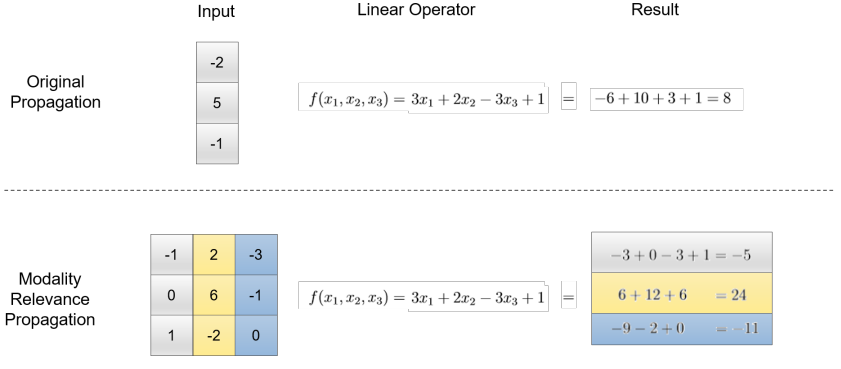

Figure S4: Visualization of an example computation of a relevance linear layer that is not the first linear layer, with bias represented in the first data source. Note that the source-wise relevance scores $(-5, 24, -11)$ add up to the total output, $-5+24-11=8$.

3.4 Proof to Proposition 1

Proposition 1 (Equivalent RFP Network)**.**
Given an input $x \in \mathcal{S}$ and corresponding prediction $f(x) \in \mathbb{R}^c$ and the prediction given by layer-wise RFP $\tilde{f}(x) \in \mathbb{R}^c$, where all layers, activation functions and operations of the neural network f are replaced by local linearizations as in eq. (3) in the main paper. Then,

$$\tilde{f}(x) = \sum_{s=0}^{n} r^s_{l=L}(x) = f(x). \tag{6}$$

Thus, for any network f consisting of locally linearizable layers, the sum of the forward propagated relevance values is equivalent to the network prediction.

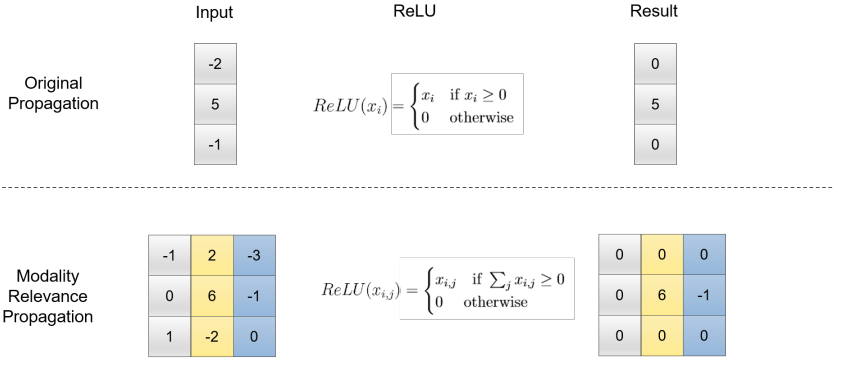

Figure S5: Visualization of an example computation of the modified ReLU function. Here, $j \in \{0,1,2\}$ denotes the data source, and the ReLU is linearized according to Lemma 1. Note that the resulting source-wise relevance scores $[(0,0,0),(0,6,0),(0,-1,0)]$ add up to the total output, $(0,0,0)+(0,6,0)+(0,-1,0) = (0,5,0)$.

Proof. The proof directly follows from the fact that all operations l_k on the input are replaced by a linearization $\tilde{l}$ as defined in Definition 2. □

3.5 Proof to Proposition 2

Proposition 2 (Equivalence of RFP and LRP Data Source Relevance Values)**.** *Let $f : \mathcal{S} \to \mathbb{R}^c$ be a neural network and fix an input $x \in \mathcal{S}$. Let $R_j^{1,s} \in \mathbb{R}$ be the relevance value of input point j of data source s, obtained by LRP applied to the m-th output $f(x)_m$, $m \in \{1,\dots c\}$, and $r_{l=L}^s(x)_m$ obtained by RFP (with weighted linearization) for the same output. Then it holds that,*

$$\sum_j R_j^{1,s} = r_{l=L}^s(x) \; . \tag{7}$$

Moreover, by adjusting the linearization $\tilde{l}$, other rules such as LRP-ε and LRP-γ can be obtained.

Proof. Wlog. we consider a single output neural network, i.e. $c = 1$, and fix an input $x \in \mathcal{S}$. This means that all relevance values backpropagated with LRP stem from this one output neuron.

Let $r_{l=L} = \left(r_{l=L}^0, r_{l=L}^1, \dots, r_{l=L}^n\right)$ the source-wise relevance score of the output neuron, received from Relevance Forward Propagation. Further, for an arbitrary neuron k in layer $l+1 \in \{1,\dots,L\}$ let R_k be the back-propagated relevance and $R_{j \leftarrow k}$ the relevance back-propagated from neuron k to neuron j at layer l, derived by the LRP back-propagation rule according to equation (16) in [1], i.e.,

$$R_{j \leftarrow k} = \frac{a_j \omega_{jk}}{\sum\limits_{l \in \mathcal{C}^-(k)} a_l \omega_{l,k}} R_k \; . \tag{8}$$

Further, we adjust the back-propagation rule for the relevance back-propagated through an RFP network to

$$R^s_{j\leftarrow k} = \frac{a^s_j \omega_{jk}}{\sum\limits_{l\in\mathcal{C}^-(k)} a_l \omega_{l,k}} R_k \,, \tag{9}$$

where a^s_j is defined by the RFP forward propagation such that $\sum_{s=0}^n a^s_j = a_j$. Precisely, for fixed input $x \in \mathcal{S}$, in the proposed Relevance Forward Propagation approach, we have source-wise outgoing activation values from each neuron k of the form $\hat{a}_k = (a^0_k, a^1_k, ..., a^n_k) \in \mathbb{R}^{n+1}$.

Two properties of (8) and (9) are important to note:

1. It is possible to apply (8) for the first few layers and then switch to (9). This is possible, as (9) only depends on the total back-propagated relevance R_k and the source-wise outgoing activations of the considered neuron j.

2. For $\sum_{s=0}^n R^s_k$, (9) is equivalent to (8), as

$$\sum_{s=0}^{n} R^s_{j\leftarrow k} = \sum_{s=0}^{n} \frac{a^s_j \omega_{jk}}{\sum\limits_{l\in\mathcal{C}^-(k)} a_l \omega_{l,k}} R_k = \frac{\sum_{s=0}^n a^s_j \omega_{jk}}{\sum\limits_{l\in\mathcal{C}^-(k)} a_l \omega_{l,k}} R_k = \frac{a_j \omega_{jk}}{\sum\limits_{l\in\mathcal{C}^-(k)} a_l \omega_{l,k}} R_k = R_{j\leftarrow k} \,.$$

Now, for V_s being the set of all input points of source s, we want to show

$$r^s_{l=L} = \sum_{p\in V_s} R_p. \tag{10}$$

We want to show that the source-wise relevance values are naturally conserved during the back-propagation from one layer to another.

To proof (10) we show two things:

1. First, we show that property (10) holds when splitting the source-wise relevance values based on the Forward Relevance Propagation and back-propagate with the adjusted backpropagation (9), working with the source-wise neuron activations $\left(a^0_k, a^1_k, ..., a^n_k\right)$ for a neuron k.

2. Second, we show that the source-wise backpropagation considered in the first point can be replaced by (8) without changing the outcome, i.e., that source-wise LRP can be replaced by the original LRP without affecting the back-propagated relevance attributed to the individual sources.

1. Backpropagation with source-wise LRP through an RFP Network

For the neuron 1in the last layer[1] $l = L$, we naturally define the source-wise relevance values based on the RFP prediction,

[1] As $c = 1$ there is only one neuron, this can be easily extended to c neurons.

$$R_1^s = r_j^s \qquad \text{and} \qquad R_1 = \sum_{s=0}^{n} R_1^s = \sum_{s=0}^{n} r_{l=L}^s = r_{l=L} \,. \tag{11}$$

For the case $R_1 = 0$, zero relevance is back-propagated to each source, and by (8) and (9), the term (10) is obvious[2]. In the following let $R_j \neq 0$, from which it follows that also $a_j \neq 0$.

Consider the fraction of the source-wise activations (computed in the forward pass) and the source-wise relevances (computed in the backward pass). For the last layer it holds for all data sources s, that

$$\frac{R_1^s}{R_1} = \frac{a_1^s}{a_1} \,. \tag{12}$$

In order to show the conservation of the source-wise relevance, we first show that (12) holds for every neuron in the network by showing that this property remains in the back-propagation process from any neuron j at layer $l+1$ in the network to connected neurons k at layer l, the previous layer, i.e., by showing

$$\frac{R_j^s}{R_j} = \frac{a_j^s}{a_j} =: \alpha_j^s \quad \Longrightarrow \quad \frac{R_k^s}{R_k} = \frac{a_k^s}{a_k} \qquad \forall\, k \in \mathcal{C}^-(j) \text{ with } R_k \neq 0 \,. \tag{13}$$

Again, the case $R_{k\leftarrow j} = 0$ is trivial, as no relevance is propagated in this case. For $R_j^s \neq 0$ [3] we see

$$R_{k\leftarrow j}^s = \frac{a_k^s \omega_{kj}}{\sum\limits_{l\in\mathcal{C}^-} a_l \omega_{lj}} R_j = \frac{a_k^s \omega_{kj}}{\sum\limits_{l\in\mathcal{C}^-} a_l \omega_{lj}} \cdot \frac{R_j^s}{\alpha_j^s} \tag{14}$$

and

$$R_{k\leftarrow j} = \frac{a_k \omega_{kj}}{\sum\limits_{l\in\mathcal{C}^-} a_l \omega_{lj}} R_j = \frac{a_k \omega_{kj}}{\sum\limits_{l\in\mathcal{C}^-} a_l \omega_{lj}} \cdot \frac{R_j^s}{\alpha_j^s} \tag{15}$$

leading to

$$\frac{R_{k\leftarrow j}^s}{R_{k\leftarrow j}} = \frac{a_k^s}{a_k} \quad \Longrightarrow \frac{R_k^s}{R_k} = \frac{\sum\limits_{j\in C^+(k)} R_{k\leftarrow j}^s}{\sum\limits_{j\in C^+(k)} R_{k\leftarrow j}} = \frac{\frac{a_k^s}{a_k} \sum\limits_{j\in C^+(k)} R_{k\leftarrow j}}{\sum\limits_{j\in C^+(k)} R_{k\leftarrow j}} = \frac{a_k^s}{a_k}, \tag{16}$$

with which (13) is shown.

With (13), we get that the property (12) holds for every neuron in the dataset, and we can show that the source-wise relevance is also conserved during the backpropagation:

$$\sum_{k\in\mathcal{C}^-(j)} R_{k\leftarrow j}^s = \frac{\sum\limits_{k\in\mathcal{C}^-(j)} a_k^s \omega_{kj}}{\sum\limits_{l\in\mathcal{C}^-(j)} a_l \omega_{lj}} R_j = \frac{a_j^s}{a_j} R_j = \alpha_j^s R_j = R_j^s$$

[2] We assume that $R_{k\leftarrow j} = 0 \implies R_{k\leftarrow j}^s = 0 \;\forall s = 0, 1, ..., n$, what is reasonable assumption for neural networks.

[3] The case $R_j^s = 0$ trivial, as this leads to $a_j^s = 0$ or $R_j = 0$)

As this property holds for all neurons by induction, it also holds for the relevance values on the input data.

Now let V_s be the set of data points of source s and let j be a neuron in the input layer. By the definition of the input mapping $\mathcal{T}$ it holds for any incoming activation of the first layer that

$$a_k^s = a_k \quad \text{if} \quad k \in V_s \qquad \text{and} \qquad a_k^s = 0 \quad \text{if} \quad k \notin V_s \, .$$

With this, it follows that for every input point k that

$$R_k^s = R_k \quad \text{if} \quad k \in V_s \qquad \text{and} \qquad R_k^s = 0 \quad \text{if} \quad k \notin V_s \, .$$

Following, we have for all $s = 0, \ldots, n$

$$\sum_{k \in \mathcal{C}^-(j) \cap V_s} R^s_{k \hookleftarrow j} = \sum_{k \in \mathcal{C}^-(j)} \frac{a_k^s \omega_{kj}}{\sum_{l \in \mathcal{C}^-(j)} a_l \omega_{lj}} R_j = \frac{a_j^s}{a_j} R_j = \alpha_j^s R_j = R_j^s$$

where the first equality holds because $a_k^s = 0$ if $k \notin V_s$. Moreover, we have for the contrary case, i.e. $k \in \mathcal{C}^-(j) \setminus V_s$ that

$$\sum_{k \in \mathcal{C}^-(j) \setminus V_s} R^s_{k \hookleftarrow j} = \sum_{k \in \mathcal{C}^-(j) \setminus V_s} \frac{0 \cdot \omega_{kj}}{\sum_{l \in \mathcal{C}^-(j)} a_l \omega_{lj}} R_j = 0 \, ,$$

where the first equality holds becuase of the definition of the input map.

Summarizing, by source-wise backpropagation through the RFP network and by the definition of the input mapping $\mathcal{T}$, we know that for the source-wise LRP

$$r^s_{l=L} = \sum_{k \in V} R_k^s = \sum_{k \in V_s} R_k^s = \sum_{k \in V_s} R_k \tag{17}$$

holds.

2. Backpropagation with normal LRP through a normal Neural Network

Finally, we can replace all backward propagation steps until the first layer from (9) to (8) without changing the relevance values in the neurons of the first layer. Most importantly, this implies that we back-propagate one scalar r_j instead of individual data source relevance values r_j^s. We also do not separate between the source-wise activations a_k^s but consider only the activation a_k. From formulas (8) and (9), it is obvious that the changes do not affect the backpropagation, as the source-wise activations and relevance values are not used iteratively but are just computed locally, while the next steps are based on the full relevance, which can also be computed based on the full activations with (8).

Most importantly, the property (17) does also still hold, i.e., also with the normal LRP backpropagation instead of the source-wise backpropagation, it holds that

$$r^s_{l=L} = \sum_{k \in V_s} R_k \tag{18}$$

with which (10) is shown. □

Extension to LRP-ε LRP-ε backpropagates the relevance with the rule

$$R_{k \leftarrow j} = \frac{a_k \omega_{kj}}{\varepsilon + \sum\limits_{l \in \mathcal{C}^-(j)} a_l \cdot \omega_{lj}}$$

for a given $\varepsilon > 0$. For an equivalent RFP-ε, the forward propagation of each neuron needs to be adjusted. For this, we consider the incoming weights ω_{lk} for all $l \in \mathcal{C}^-(k)$ and want to adjust them such that the resulting RFP-ε forward step is equivalent to an LRP-ε backward pass. The adjusted weights $\tilde{\omega}_{lk}$ have to fulfil two properties:

1. The sum of all incoming values shall not change, i.e.,

$$\sum_{l \in \mathcal{C}^-(j)} \sum_{s^0}^{n} a^s_l \tilde{\omega}_{lk} \overset{!}{=} \sum_{l \in \mathcal{C}^-(j)} \sum_{s=0}^{n} a^s_l \omega_{lj}$$

2. The LRP-ε rule needs to be fulfilled, i.e.,

$$\frac{a_k \tilde{\omega}_{kj}}{\sum\limits_{l \in \mathcal{C}^-(j)} a_l \cdot \tilde{\omega}_{lj}} \overset{!}{=} \frac{a_k \omega_{kj}}{\varepsilon + \sum\limits_{l \in \mathcal{C}^-(j)} a_l \cdot \omega_{lj}}$$

Let $z_j = \sum_{l \in \mathcal{C}^-(j) \setminus \{b\}} \omega_{lj} a_l$ and z^s_j the source-wise version respectively, where the sum is without the bias term b_j. We make sure that the two conditions are fulfilled by shifting a little relevance from all incoming activations to the bias of the considered neuron, i.e., we adjust the bias b to

$$\tilde{b}_j = b_j + \frac{z_j \cdot \varepsilon}{z_j + \varepsilon}$$

and all other incoming weights to

$$\tilde{\omega}_{lj} = \omega_{lj} \frac{z_j}{z_j + \varepsilon} \,.$$

With this, both conditions are fulfilled:

1.

$$\begin{aligned}
\sum_{l \in \mathcal{C}^-(j)} \sum_{s^0}^{n} a^s_l \tilde{\omega}_{lj} &= \sum_{l \in \mathcal{C}^-(j) \setminus \{b\}} \sum_{s^0}^{n} a^s_l \tilde{\omega}_{lj} + \sum_{s^0}^{n} \tilde{b}^s_j \\
&= \sum_{l \in \mathcal{C}^-(j) \setminus \{b\}} \sum_{s^0}^{n} a^s_l \omega_{lj} \frac{z_j}{z_j + \varepsilon} + b_j + \frac{z_j \cdot \varepsilon}{z_j + \varepsilon} \\
&= z_j \frac{z_j}{z_j + \varepsilon} + b_j + \frac{z_j \cdot \varepsilon}{z_j + \varepsilon} \\
&= \sum_{l \in \mathcal{C}^-(j)} \sum_{s=0}^{n} a^s_l \omega_{lj}
\end{aligned}$$

2.

$$\frac{a_l\tilde{\omega}_{lj}}{\sum\limits_{l\in\mathcal{C}^-(j)} a_l\cdot\tilde{\omega}_{lj}} \overset{1.}{=} \frac{a_l\tilde{\omega}_{lj}}{\sum\limits_{l\in\mathcal{C}^-(j)} a_l\cdot\omega_{lj}} = \frac{a_l\omega_{lj}\frac{z_j}{z_j+\varepsilon}}{z_j} = \frac{a_l\omega_{lj}}{z_j+\varepsilon} = \frac{a_l\omega_{lj}}{\varepsilon+\sum\limits_{l\in\mathcal{C}^-(j)} a_l\cdot\omega_{lj}}$$

RFP-γ Adjust all incoming weights of a neuron j to

$$\tilde{\omega}_{kj} = \omega_{kj}\cdot(1+\gamma\cdot\omega_{kj}^+)\frac{\sum_{l\in\mathcal{C}^-(j)} a_l\omega_{lj}}{\sum\limits_{l\in\mathcal{C}^-(j)} a_l\omega_{lj}\cdot\left(1+\omega_{lj}^+\right)}\,.$$

The following two properties have to be fulfilled:

1. The sum of all incoming values shall not change, i.e.,

$$\sum_{l\in\mathcal{C}^-(j)}\sum_{s^0}^{n} a_l^s\tilde{\omega}_{lk} \overset{!}{=} \sum_{l\in\mathcal{C}^-(j)}\sum_{s=0}^{n} a_l^s\omega_{lj}$$

2. The LRP-γ rule needs to be fulfilled, i.e.,

$$\frac{a_k\tilde{\omega}_{kj}}{\sum\limits_{l\in\mathcal{C}^-(j)} a_l\cdot\tilde{\omega}_{lj}} \overset{!}{=} \frac{a_k\left(\omega_{lj}+\gamma\omega_{lj}^+\right)}{\sum\limits_{l\in\mathcal{C}^-(j)} a_l\cdot\left(\omega_{lj}+\gamma\omega_{lj}^+\right)}$$

Both properties directly follow from plugging in the definition of $\tilde{\omega}_{lj}$.

4 Datasets

4.1 Multi-Source MNIST

The different types of the Multi-Source MNIST were all applied to a simple network based on two input branches consisting of two convolutional layers with 32 and 64 filters of size 3, followed by two joint linear layers with an inner dimension of 512. The training is run for 10 epochs (25 epochs for the summation case) with the stochastic gradient descent optimizer and a learning rate of 0.1.

4.1.1 Multi-Source MNIST - Basic

For the basic version of the Multi-Source MNIST, it is interesting to see how the level of contribution differs between the different classes. For example, class 8 is easy to confuse with a 3 if only the right part of the image is considered.

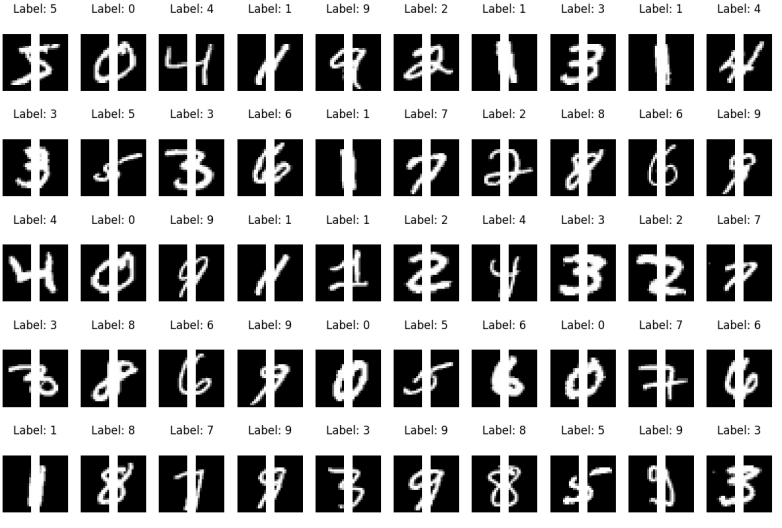

Figure S6: Examples of the basic Multi-Source MNIST dataset.

4.1.2 Multi-Source MNIST - Summation

In contrast to the other cases, we trained this setup for 25 epochs and randomly sampled the left and right sides for each input from the whole test set.

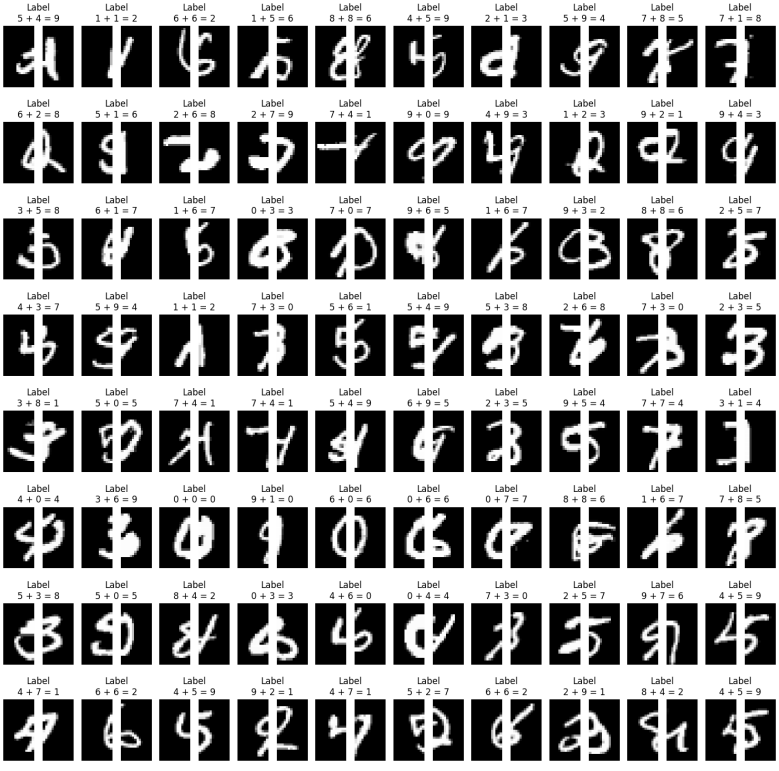

Figure S7: Examples of the Label Summation Multi-Source MNIST dataset.

4.1.3 Multi-Source MNIST - Label Noise

For the label noise case, the network is supposed to take more information from the non-noisy data source. As the network can not detect the difference between a correct and a false label, it is supposed to also focus significantly more on the non-noisy source.

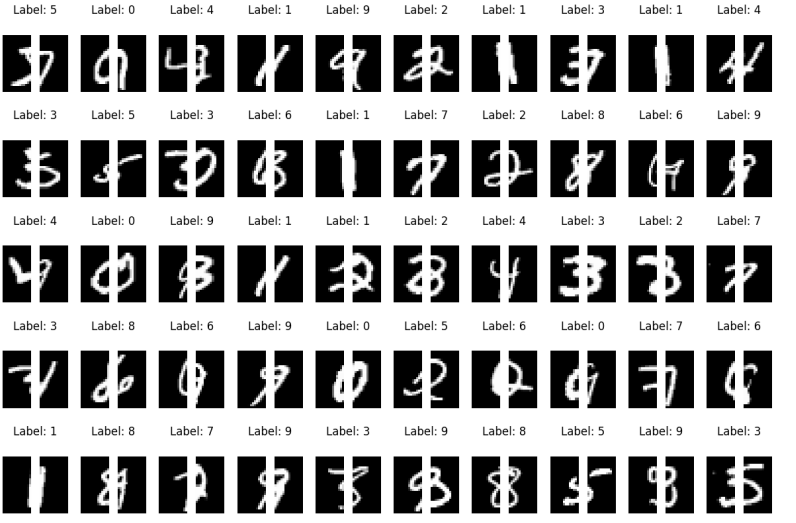

Figure S8: Examples of the Label Noise Multi-Source MNIST dataset.

4.1.4 Multi-Source MNIST - Data Noise

For the data noise case, the network is supposed to take more information from the non-noisy or less noisy data source. In contrast to the label noise case, the network will learn to differentiate between noisy (ignore them) and non-noisy (include them) samples.

Figure S9: Examples of the Data Noise Multi-Source MNIST dataset.

4.1.5 Multi-Source MNIST - Pixel Shuffle

For the pixel shuffle case, the network extracts more information from the data source containing more information (the unshuffled one). Depending on the level of information content left, the shuffled data source can (in theory) still contain valuable information.

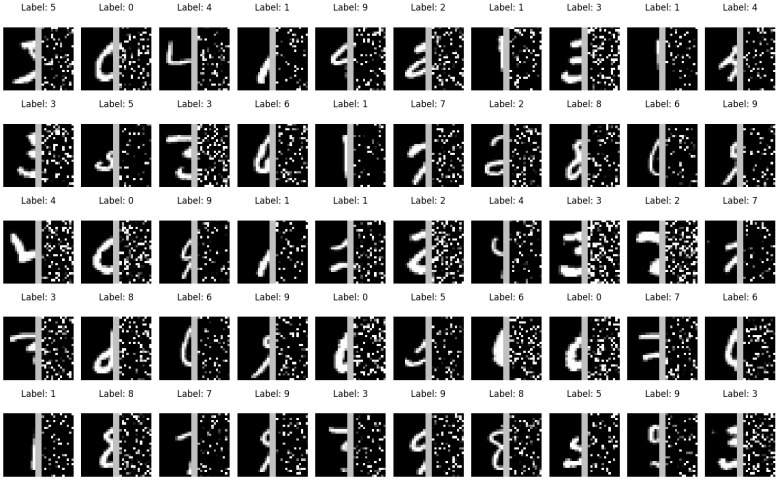

Figure S10: Examples of the Pixel-Shuffle Multi-Source MNIST dataset.

4.2 SEN12MS Dataset

The SEN12MS dataset [12] comprises 180,662 co-registered images from Sentinel-1 and Sentinel-2 satellites. The images have a resolution of 256 × 256 pixels and are taken from 252 regions of interest distributed globally. The optical Sentinel-2 samples in the dataset consist of 13 multi-spectral channels. All samples are free of clouds and have undergone atmospheric correction. The Sentinel-1 patches in the dataset include the two polarimetric channels (VV and VH). Each pixel is assigned to one of 10 land cover classes using the MODIS-derived [4] simplified IGBP scheme [12].

4.2.1 Classification on SEN12MS

Data For the classification task, the class labels are determined through majority voting of pixel-wise MODIS labels [11]. For testing on cloudy samples, we used the split provided in [6], which consists of co-registered cloudy, non-cloudy, and Sentinel-1 samples. The split includes cloud coverage ranging from very little to full cloud coverage. The pre-trained model achieves an accuracy value of 0.646636 on the clear test set and 0.436375 on the corresponding cloudy dataset.

Model We apply the Relevance Forward Propagation on the pre-trained ResNet50 model provided by the authors of the dataset [11]. We do not apply any finetuning or modification other than the Relevance Forward Propagation adaptation. The network uses early fusion and expects the input to concatenate ten of the 13 optical bands and the two SAR channels.

4.2.2 Weakly Supervised Segmentation on SEN12MS

For the pixel-wise segmentation task, each pixel is assigned to one of 10 land cover classes using the MODIS-derived [4] simplified IGBP scheme [12].
We applied the Relevance Forward Propagation to pre-trained versions of the DeepLabV3+ model without any finetuning or other modifications than the presented source-wise relevance separation. For the evaluation of the weakly supervised segmentation, we built up on the baseline repository of the data fusion contest 2020 (https://github.com/lukasliebel/dfc2020_baseline).

4.3 ADVANCE Dataset

The AuDio Visual Aerial sceNe reCognition datasEt (ADVANCE) [7] is a multimodal dataset designed to enhance scene recognition by integrating audio and visual data. It comprises 5075 pairs of geotagged aerial images and corresponding sounds, classified into 13 scene categories, including airports, beaches, forests, and residential areas. The audio data, sourced from Freesound [3] and paired with aerial images from Google Earth, have been curated and annotated using OpenStreetMap coordinates, with manual verification for accuracy. To address the imbalance in scene classes, scenes with fewer samples were either excluded or augmented by generating additional aerial images from slightly varied geographic coordinates, ensuring a more balanced dataset for training purposes. This dataset aims to advance research in aerial scene understanding and audiovisual learning.

5 Supplemental Results

5.1 Multi-Source MNIST Basic

Fig. S12 shows examples of the basic MNIST version.

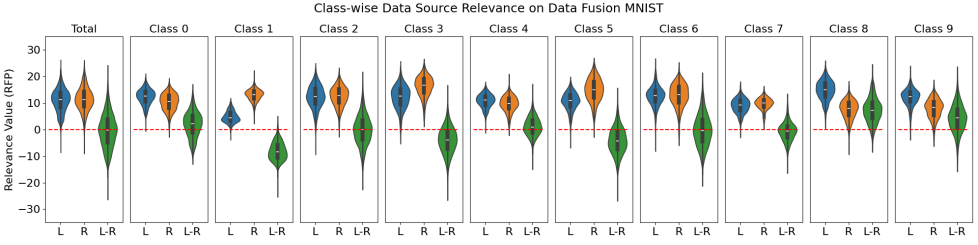

Figure S11: Class-wise distribution of the source-wise relevance values received via Relevance Forward Propagation for a network trained to predict the Basic Multi-Source MNIST Dataset.

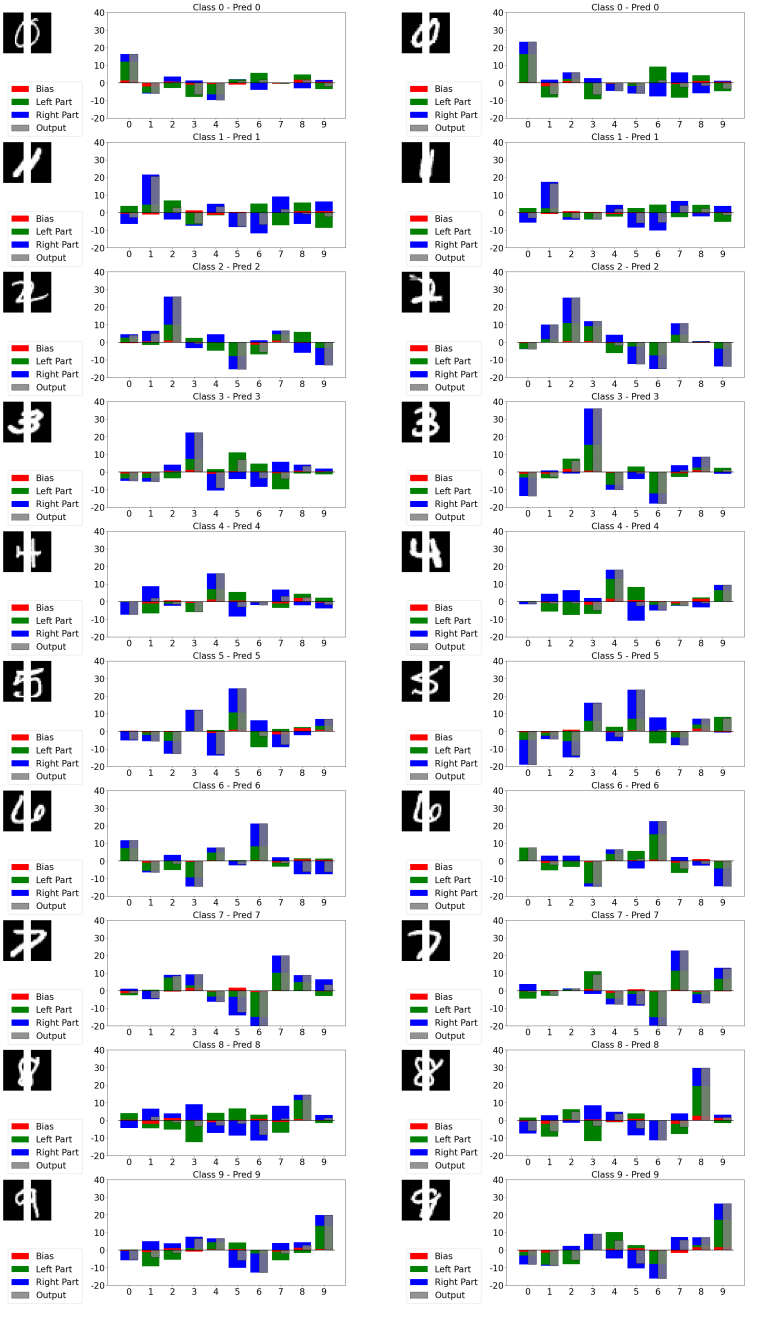

Figure S12: Examples for the Relevance Forward Propagation applied to the Basic Multi-Source MNIST Dataset.

5.2 Multi-Source MNIST with Label Summation

Fig. S13 shows the class-wise distribution of the source-wise relevance values received via Relevance Forward Propagation for a network trained to predict the sum modulo 10 of the two input sources.

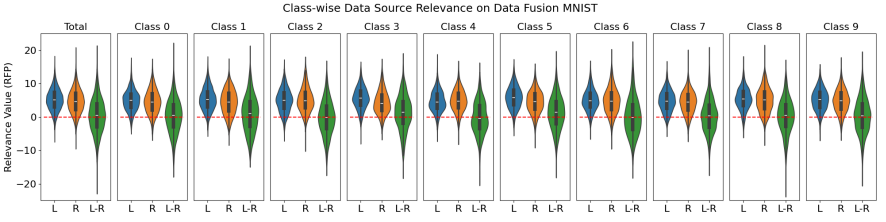

Figure S13: Class-wise distribution of the source-wise relevance values received via Relevance Forward Propagation for a network trained to predict the sum modulo 10 of the two input sources.

5.3 Multi-Source MNIST with Label Noise

Fig. S14 and Fig. S16 show the distributions of the source-wise RFP relevance values when trained with label noise on the first and the second data source. The label noise clearly affects the relevance of the noisy labeled data source.

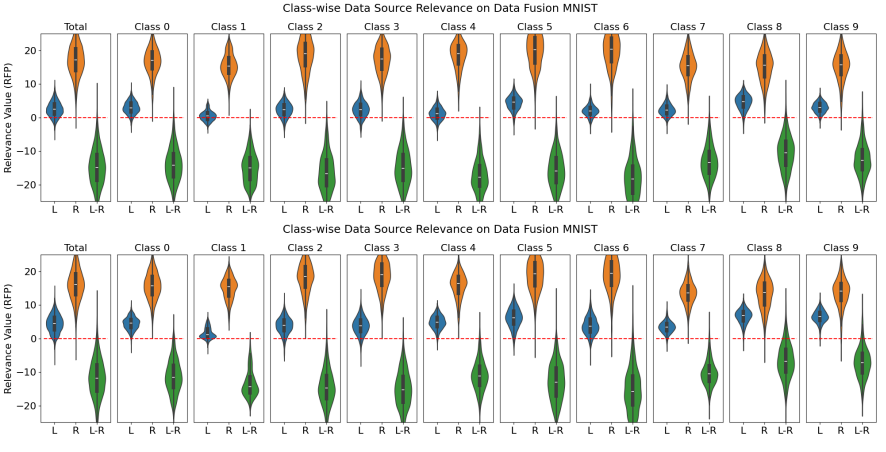

Figure S14: Class-wise distribution of the source-wise relevance values received via Relevance Forward Propagation. The Network is trained with 20% (top) and 5% (bottom) label noise on the training samples from data source 1. One can clearly see that the overall relevance of the noisy labeled data source is smaller.

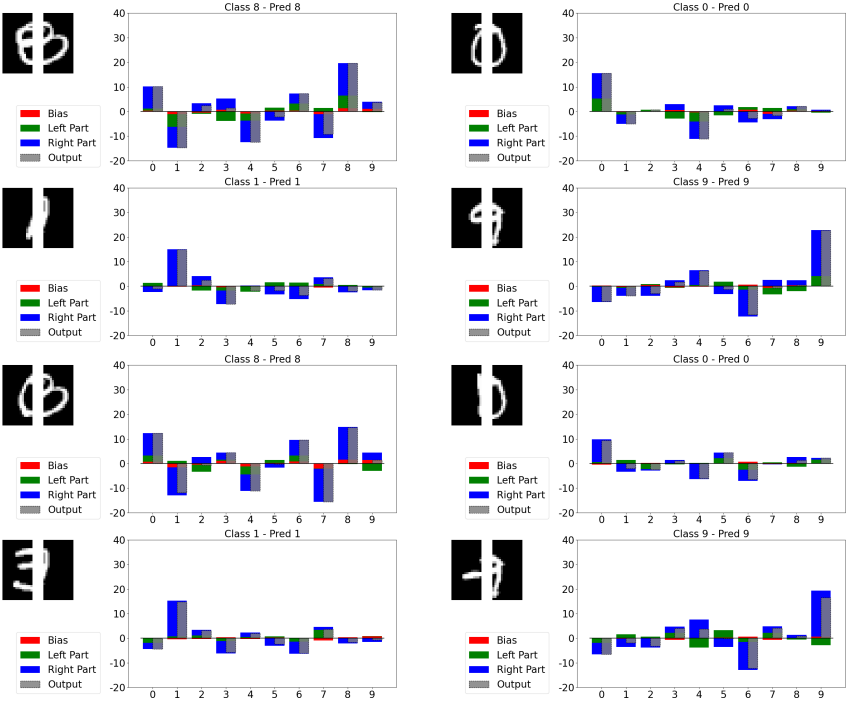

Figure S15: Example predictions with 20% label noise in the first data source in the training. The first two rows show test samples without label noise, and the lower two rows show test samples with label noise in the first data source.

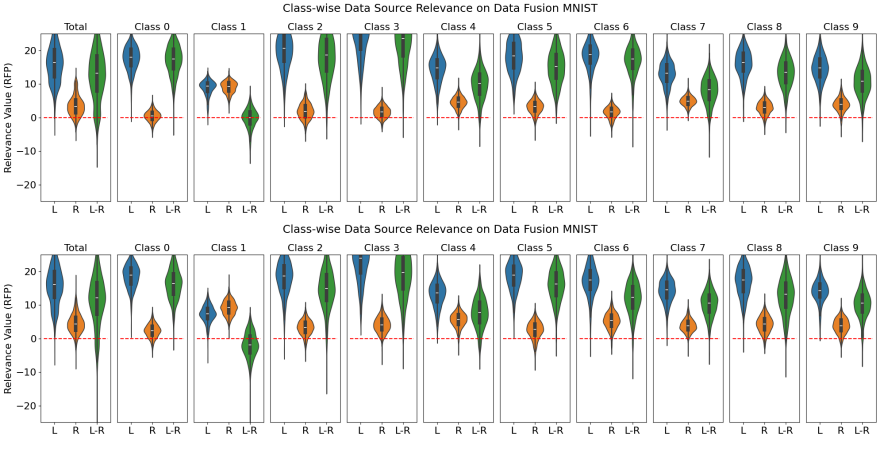

Figure S16: Class-wise distribution of the source-wise relevance values received via Relevance Forward Propagation. The Network is trained with 20% (top) and 5% (bottom) label noise on the training samples from data source 2. One can clearly see that the overall relevance of the noisy labeled data source is smaller.

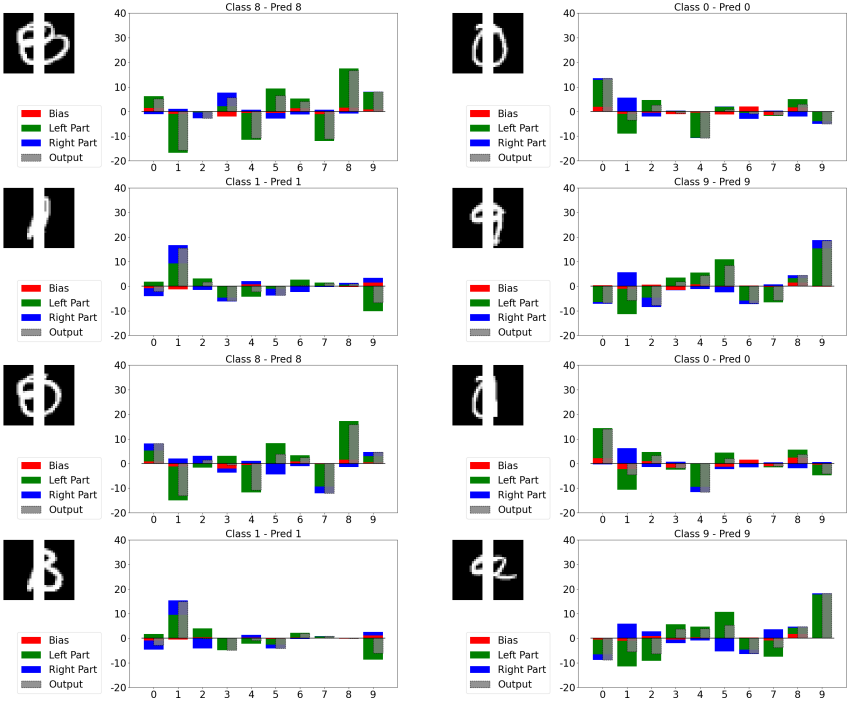

Figure S17: Example predictions with 20% label noise in the first data source in the training. The first two rows show test samples without label noise, and the lower two rows show test samples with label noise in the first data source.

5.4 Multi-Source MNIST with Data Noise

Fig. S18, Fig. S20 and Figure Fig. S22 show the distributions of the source-wise RFP relevance values when trained with data noise on the first, the second or on both data source and tested with and without data noise. The data noise clearly affects the relevance of the noisy data source, and the network learns to neglect information from noisy examples.

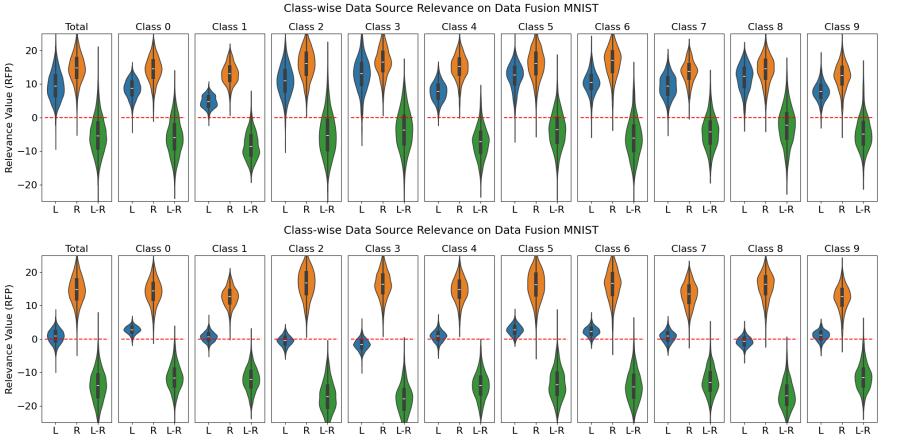

Figure S18: Class-wise distribution of the source-wise relevance values received via Relevance Forward Propagation. The network is trained with 20% noisy samples in data source 1 and tested with 0% noisy samples (top) and 100% noisy samples (bottom). One can clearly see that the overall relevance of the noisy data source is smaller, even for non-noisy samples. For noisy samples, the network learned to ignore the noise.

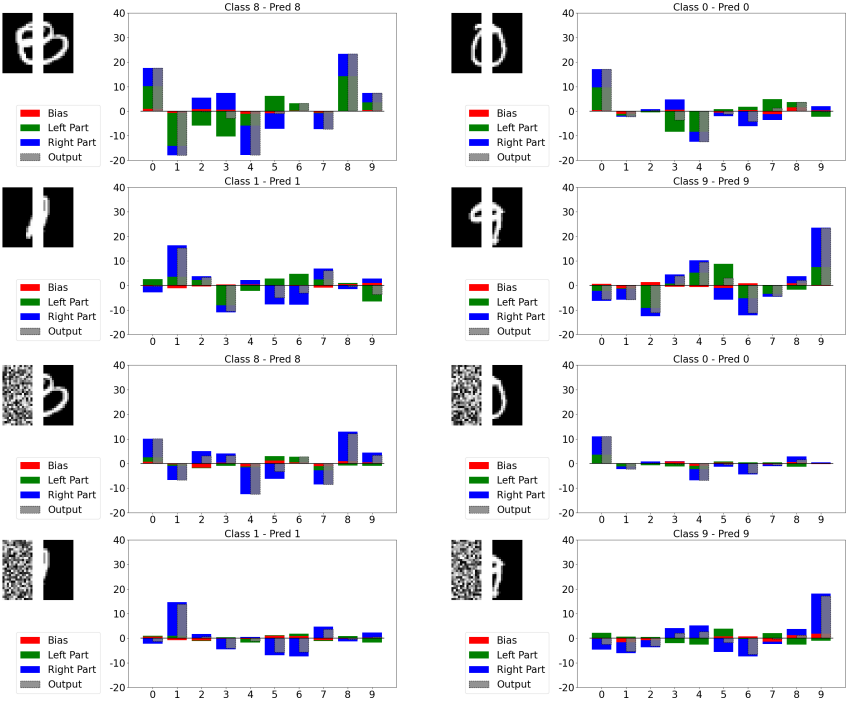

Figure S19: Example predictions with 20% data noise in the first data source in the training. The first two rows show test samples without data noise, and the lower two rows show test samples with data noise in the first data source.

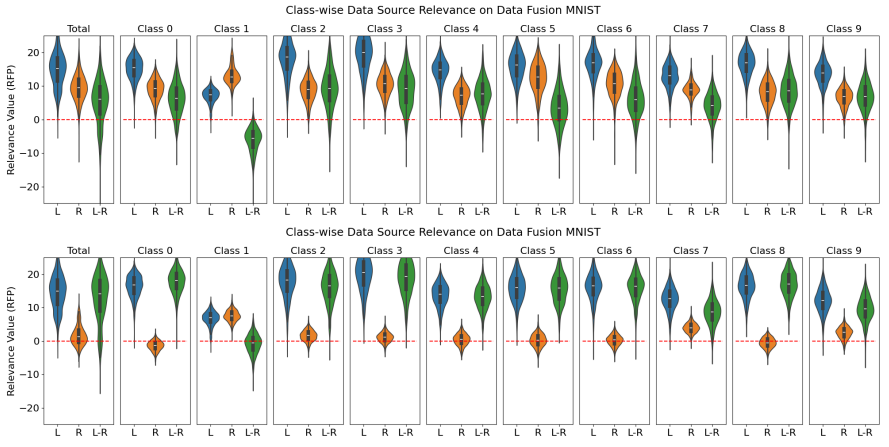

Figure S20: Class-wise distribution of the source-wise relevance values received via Relevance Forward Propagation. The network is trained with 20% noisy samples in data source 2 and tested with 0% noisy samples (top) and 100% noisy samples (bottom). One can clearly see that the overall relevance of the noisy data source is smaller, even for non-noisy samples. For noisy samples, the network learned to ignore the noise.

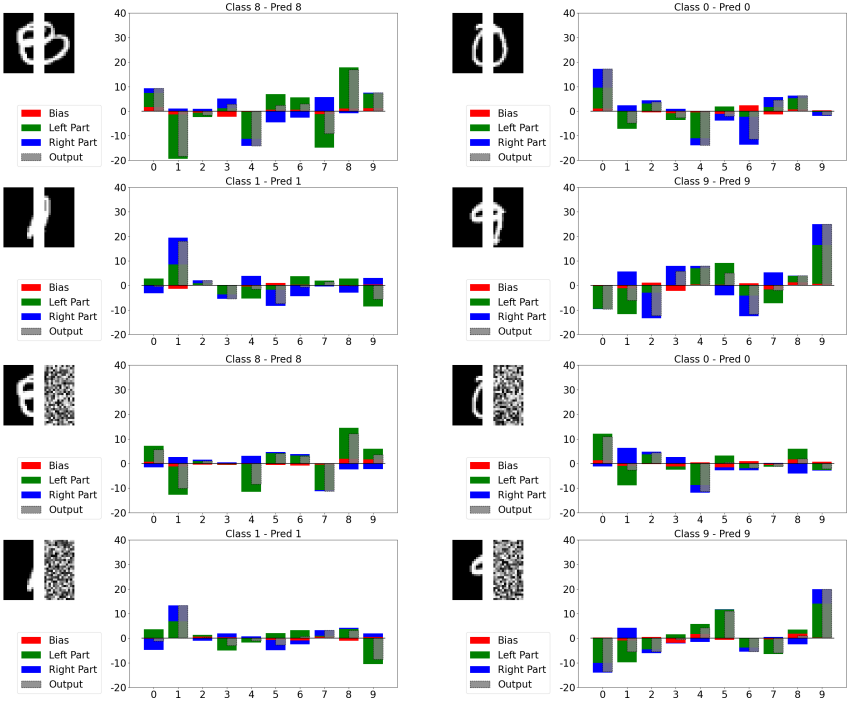

Figure S21: Example predictions with 20% data noise in the first data source in the training. The first two rows show test samples without data noise, and the lower two rows show test samples with data noise in the first data source.

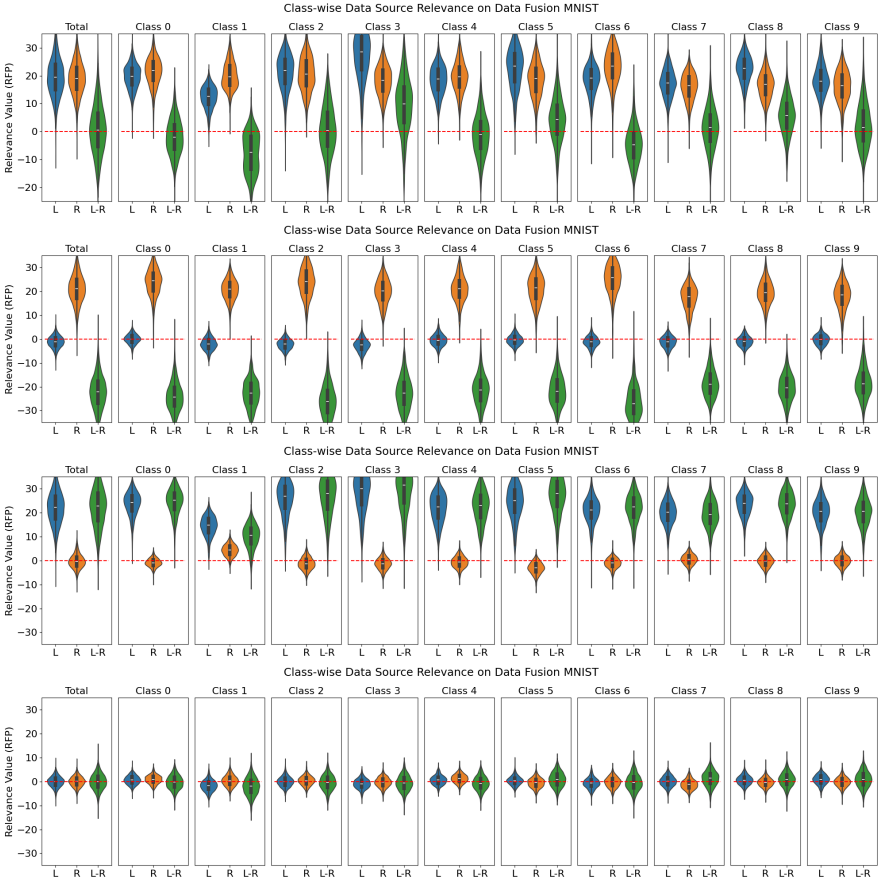

Figure S22: Class-wise distribution of the source-wise relevance values received via Relevance Forward Propagation. The network is trained with 20% noisy samples in data source 1 and data source 2 and tested with 0% noisy samples (top) and 100% noisy samples in data source 1 (2nd row), 100% noisy samples in data source 2 (3rd row) and 100% noisy samples in both data sources (bottom). One can clearly see that the overall relevance of the noisy data source is smaller, even for non-noisy samples. For noisy samples, the network learned to ignore the noise.

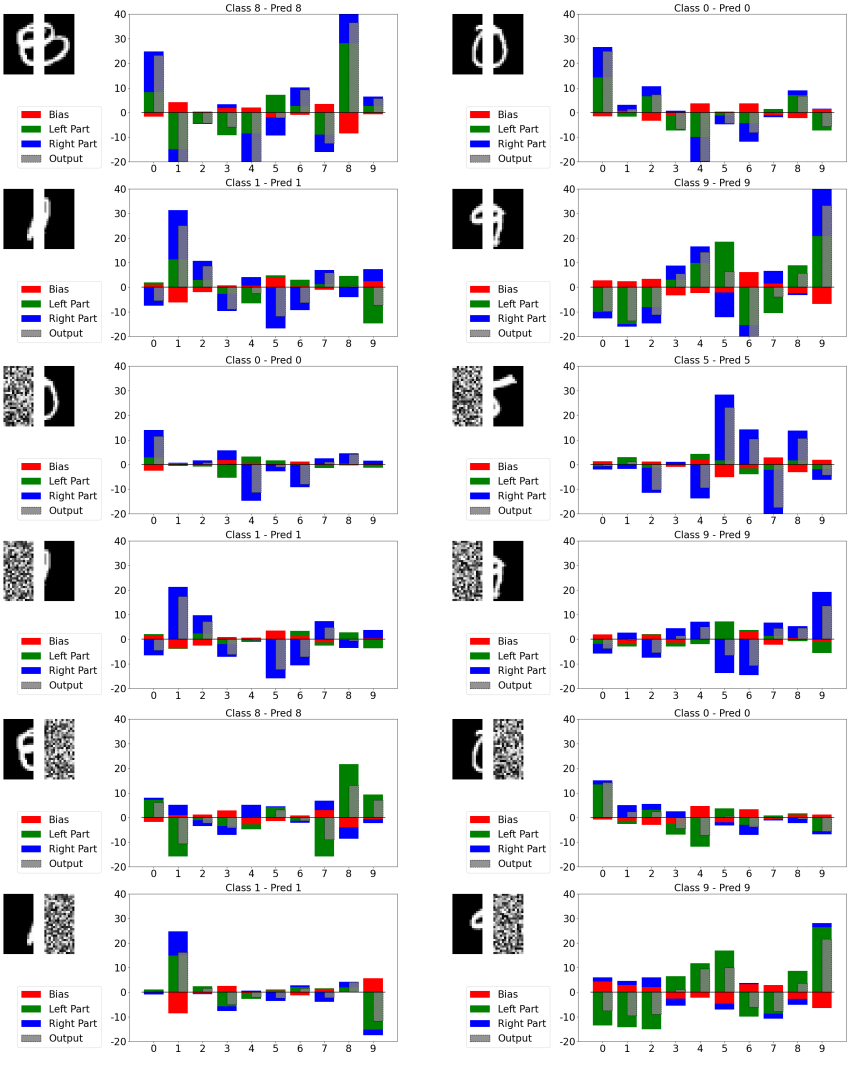

Figure S23: Example predictions with 20% data noise in both data sources in the training. The first two rows show test samples without data noise, the second two rows show test samples with data noise in the first data source, followed by two rows with data noise in the second data source.

5.5 Multi-Source MNIST with Shuffled Pixels

Fig. S24 shows the class-wise data source relevance received from Relevance Forward Propagation.

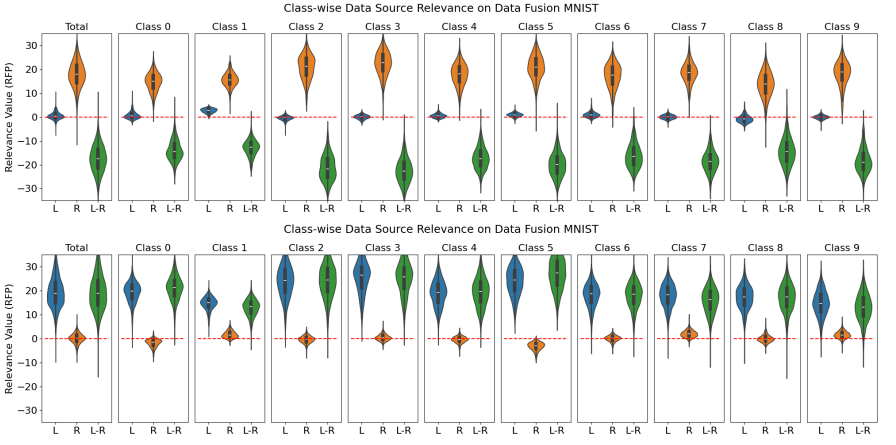

Figure S24: Distributions over the sample-wise relevance given by RFP and the Perceptual score when the pixels of the first data source (top) and the second data source (bottom) are shuffled.

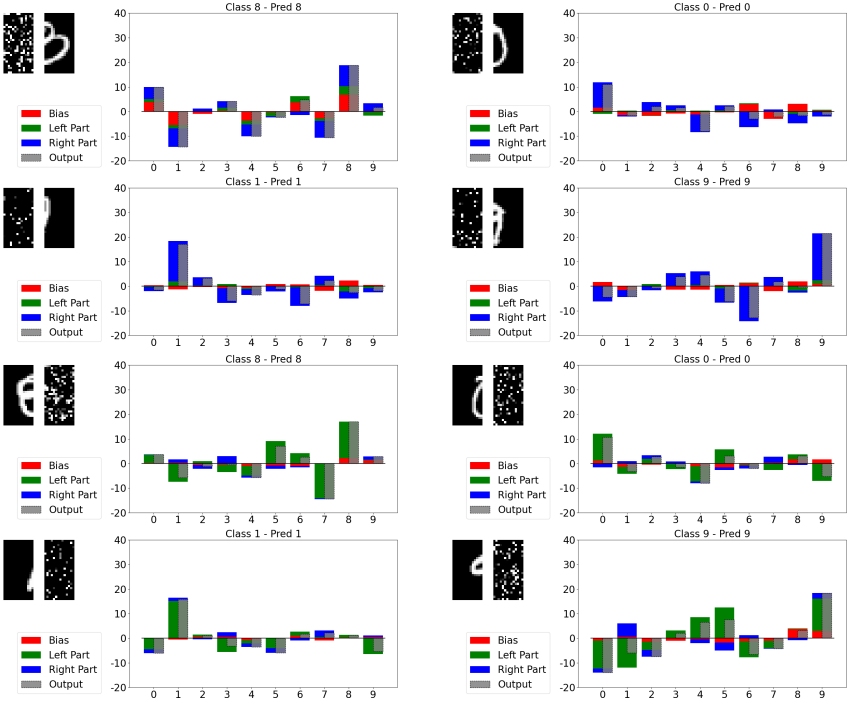

Figure S25: Example predictions with shuffled pixels in the first data source (top) and the second data source (bottom).

5.6 Multi-Source MNIST Types Comparison

Fig. S26 and Tab. S1 show the distributions of the source-wise RFP relevance values and Perceptual Scores for the different setups of the multi-source MNIST.

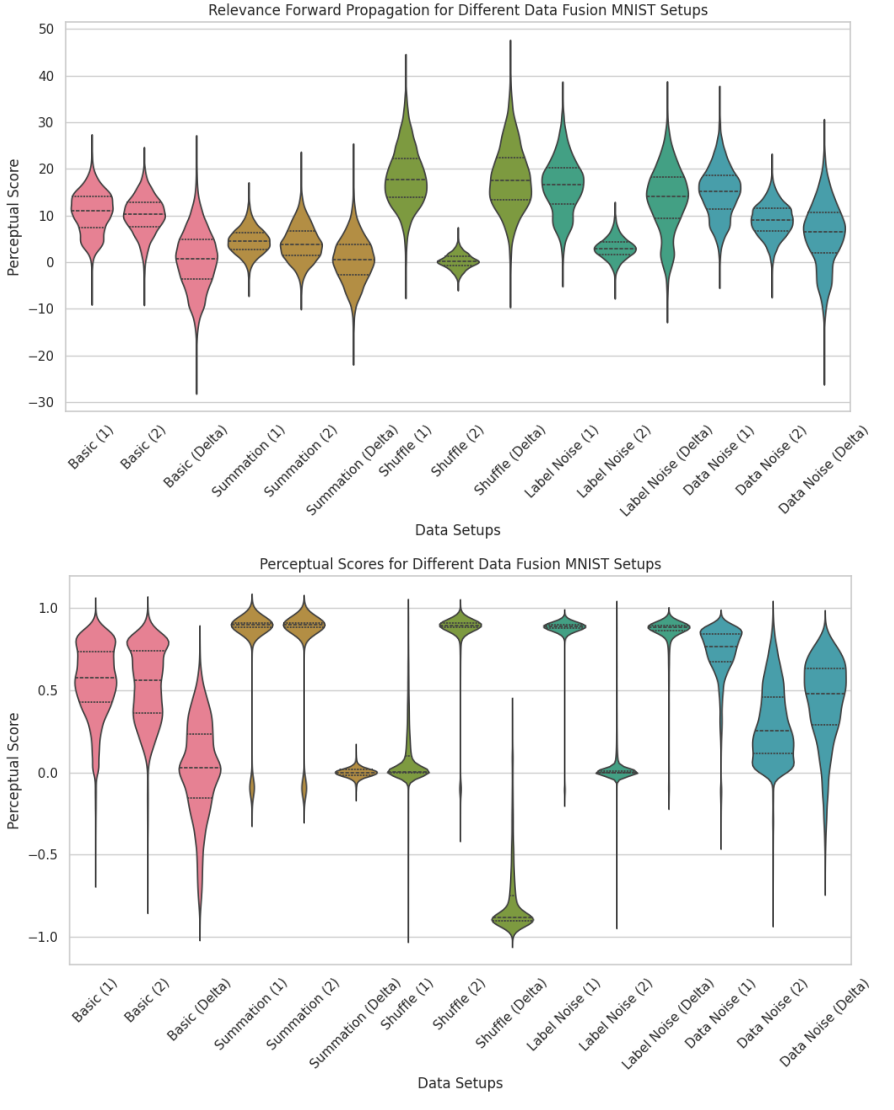

Figure S26: Distributions over the sample-wise relevance given by RFP and the Perceptual score. The main trends among the use cases are the same for the two approaches. Only for the summation use case is the need for both modalities more visible in the perceptual score.

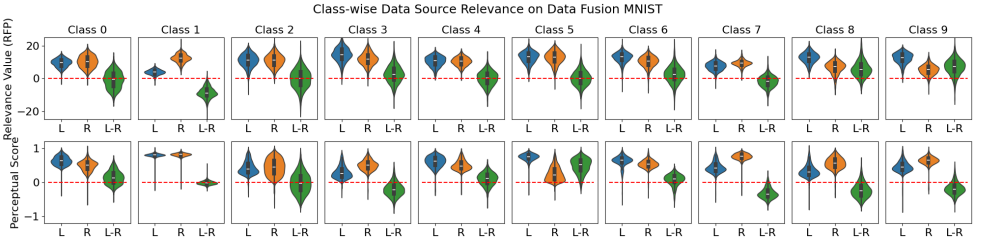

Figure S27: Class-wise distributions of the relevance scores derived by RFP (first row) and Perceptual Score (second row). For each class, the relevance of the left part (L), the right part (R), and the sample-wise difference (L-R) are given.

		Basic	Summation	Pixel Shuffle	Data Noise	Label Noise
RFP	Left	10.69±4.48	4.56±2.79	18.26±6.23	16.41±5.96	14.99±5.42
	Right	10.06±4.05	4.10±3.84	0.20±1.59	2.99±2.26	8.99±3.56
	Delta	0.63±6.35	0.46±4.90	18.06±6.85	13.42±7.05	6.00±6.81
PS	Left	0.56±0.22	0.79±0.31	0.86±0.19	0.87±0.13	0.73±0.17
	Right	0.54±0.23	0.80±0.31	0.08±0.19	0.04±0.14	0.29±0.23
	Delta	0.02±0.29	0.00±0.03	0.78±0.23	0.83±0.17	0.43±0.26
SHAPE	Left	0.43	0.40	0.85	0.83	0.54
	Right	0.46	0.40	0.02	0.05	0.35
	Delta	−0.03	0.00	0.83	0.78	0.19
G-CAM	Left	0.71±0.81	0.40±0.38	1.08±0.82	0.51±0.49	0.57±0.51
	Right	0.18±0.18	0.43±0.39	0.07±0.06	0.18±0.15	0.09±0.07
	Delta	0.53±0.78	−0.03±0.52	1.01±0.81	0.33±0.46	0.48±0.49

Table S1: Source-wise relevance of the left and the right components of the different versions of the multi-source MNIST dataset. The relevance values from RFP are evaluated for the predicted class, and the Perceptual Score (PS) is computed with respect to the true class and predictions. SHAPE is stated without standard deviation, as the computation is only possible over the full dataset. Delta describes the sample-wise difference in the relevance values (left-right).

5.7 SEN12MS - Classification

Fig. S28 and Fig. S29 show examples of clear and cloudy predictions and relevance values computed with Forward Relevance Propagation.

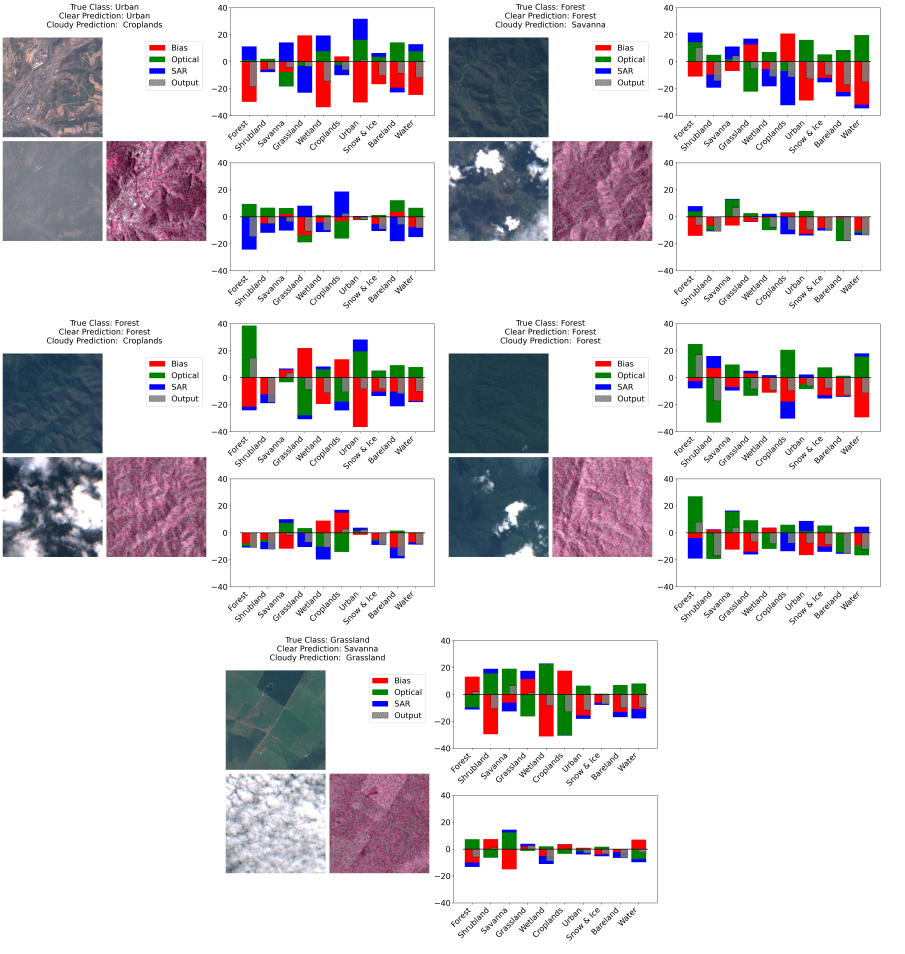

Figure S28: Examples of clear and cloudy predictions and relevance values computed with Forward Relevance Propagation.

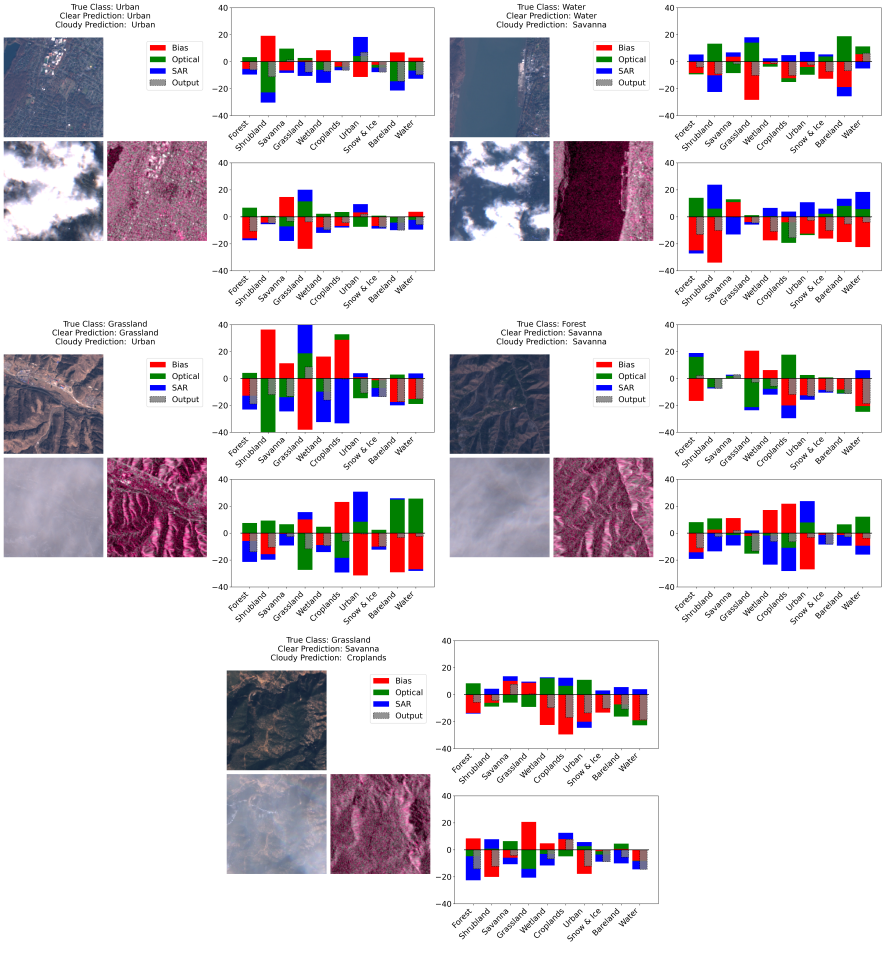

Figure S29: Examples of clear and cloudy predictions and relevance values computed with Forward Relevance Propagation.

		SAR	Optical	Delta
RFP	Clear	$-0.29_{\pm 8.59}$	$+4.45_{\pm 11.95}$	$-4.74_{\pm 14.91}$
	Cloudy	$+1.00_{\pm 8.40}$	$-0.29_{\pm 10.20}$	$+1.29_{\pm 13.70}$
PS	Clear	$+0.14_{\pm 0.35}$	$+0.41_{\pm 0.35}$	$-0.27_{\pm 0.31}$
	Cloudy	$+0.10_{\pm 0.34}$	$+0.23_{\pm 0.34}$	$-0.13_{\pm 0.35}$
GC	Clear	$+0.26_{\pm 0.54}$	$+1.01_{\pm 1.96}$	$+0.74_{\pm 1.44}$
	Cloudy	$+0.29_{\pm 0.54}$	$+1.19_{\pm 2.05}$	$+0.90_{\pm 1.55}$

Table S2: SAR, Optical, and Delta (sample-wise SAR minus Optical) source-wise relevance values on the clear and cloud-containing test set. The values are computed with Relevance Forward Propagation (RFP), Perceptual Score (PS), SHAPE (SH), and Grad-CAM (GC). The relevance is evaluated for the predicted class for Grad-CAM and RFP. The Perceptual Score and SHAPE are computed with respect to the true class. One can clearly see that the optical component's relevance drops when cloudy samples have not been replaced. Delta describes the sample-wise difference in the relevance values (SAR-Optical). SHAPE is stated without standard deviation, as the computation is only possible over the full dataset.

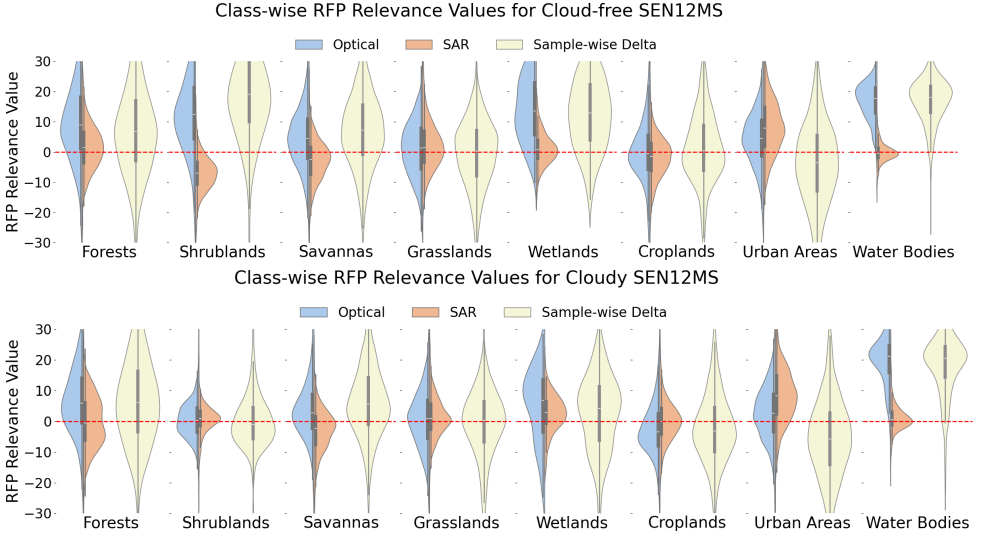

Figure S30: Class-wise Distribution of RFP relevance values on the SEN12MS test set without clouds (top) and including cloudy samples (bottom). Sample-wise, Delta describes the optical optical relevance minus the SAR relevance values.

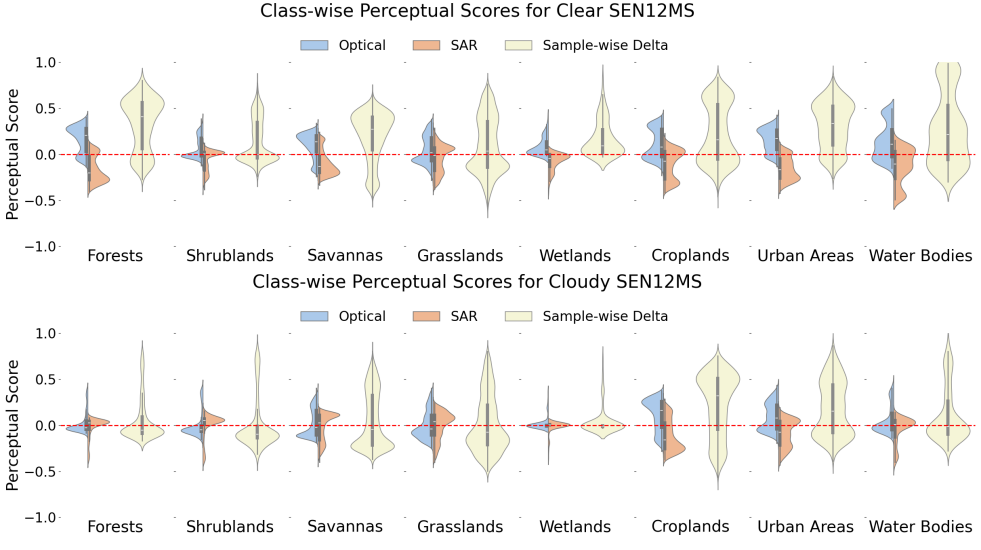

Figure S31: Class-wise Distribution of Perceptual Scores on the SEN12MS test set without clouds (top) and including cloudy samples (bottom). Sample-wise, Delta describes the optical optical relevance minus the SAR relevance values.

5.8 ADVANCE

In Tab. S3, the relevance values for the audio and the optical modality are shown. As stated in the main paper, it is clearly visible in all considered approaches that the optical component is more relevant, and the relevance of the additional audio modality is minor. As the data source

		Total	Airport	Beach	Bridge	Farm-land	Forest	Grass-land	Harbour	Lake	Or-chard	R. Area	Delta	Shrub Land	Train Station
RFP	Optical	$0.67_{\pm 17.42}$	$-6.66_{\pm 24.26}$	$-0.11_{\pm 15.64}$	$1.66_{\pm 15.72}$	$3.46_{\pm 14.26}$	$-0.32_{\pm 17.10}$	$3.63_{\pm 17.92}$	$0.93_{\pm 12.02}$	$-1.48_{\pm 10.80}$	$-0.34_{\pm 8.26}$	$6.25_{\pm 20.11}$	$3.59_{\pm 18.86}$	$0.90_{\pm 23.26}$	$-1.51_{\pm 16.03}$
	Audio	$0.01_{\pm 0.23}$	$-0.19_{\pm 0.38}$	$0.08_{\pm 0.21}$	$0.03_{\pm 0.34}$	$0.24_{\pm 0.21}$	$-0.15_{\pm 0.15}$	$0.01_{\pm 0.14}$	$-0.02_{\pm 0.15}$	$-0.13_{\pm 0.16}$	$0.08_{\pm 0.19}$	$0.03_{\pm 0.27}$	$0.09_{\pm 0.17}$	$0.05_{\pm 0.17}$	$-0.02_{\pm 0.23}$
PS	Optical	$0.62_{\pm 0.42}$	$0.48_{\pm 0.50}$	$0.73_{\pm 0.23}$	$0.62_{\pm 0.45}$	$0.45_{\pm 0.50}$	$0.76_{\pm 0.40}$	$0.71_{\pm 0.39}$	$0.70_{\pm 0.31}$	$0.69_{\pm 0.43}$	$0.40_{\pm 0.49}$	$0.73_{\pm 0.40}$	$0.27_{\pm 0.45}$	$0.50_{\pm 0.49}$	$0.43_{\pm 0.50}$
	Audio	$0.01_{\pm 0.08}$	$0.03_{\pm 0.11}$	$0.00_{\pm 0.00}$	$0.01_{\pm 0.09}$	$0.02_{\pm 0.11}$	$0.02_{\pm 0.11}$	$0.00_{\pm 0.04}$	$0.00_{\pm 0.03}$	$-0.01_{\pm 0.12}$	$0.02_{\pm 0.11}$	$0.04_{\pm 0.17}$	$0.00_{\pm 0.00}$	$0.01_{\pm 0.09}$	$0.00_{\pm 0.06}$
SH	Optical	0.66	0.60	0.96	0.72	0.41	0.82	0.29	0.88	0.71	0.37	0.73	−0.13	0.58	0.44
	Audio	0.03	−0.02	0.00	0.01	0.00	0.00	0.50	0.01	0.00	0.00	0.00	−0.49	−0.01	0.00

Table S3: Source-wise relevance of the optical and the audio source of the test split of the ADVANCE dataset. The audio component has very little relevance according to all approaches.

relevance distribution shows a high bias towards the optical data. However, we followed the official pipelines and the provided pre-implementations from the authors' GitHub repository and received comparable performance values with an overall accuracy of 74.48.

5.9 SEN12MS - Segmentation

Fig. S32 and Fig. S33 show examples of the resulting relevance maps with respect to the predicted class. In most cases, a clear edge between the SAR input and the optical input is visual and is caused by a switch in the predicted label, i.e., the two modalities are differently relevant to the shown classes, and hence, the shown relevance changes together with the predicted class. In Fig. S34, the bias contributions are split among the optical and the SAR modalities. The split is realized based on the individual relevance of the two modalities. For a scalar linear transformation with weight $w \in \mathbb{R}$ and bias $b \in \mathbb{R}$ this would transform

$$a \cdot x + b \qquad \text{to} \qquad a \cdot (x_{\text{opt}}, x_{\text{sar}}) + \frac{(x_{\text{opt}} + \varepsilon, x_{\text{sar}} + \varepsilon)}{|x_{\text{opt}}| + |x_{\text{sar}}| + 2\varepsilon} \cdot b \,.$$

Please note that the case that the enumerator is zero is rather unlikely to occur in neurons with higher input dimensions.

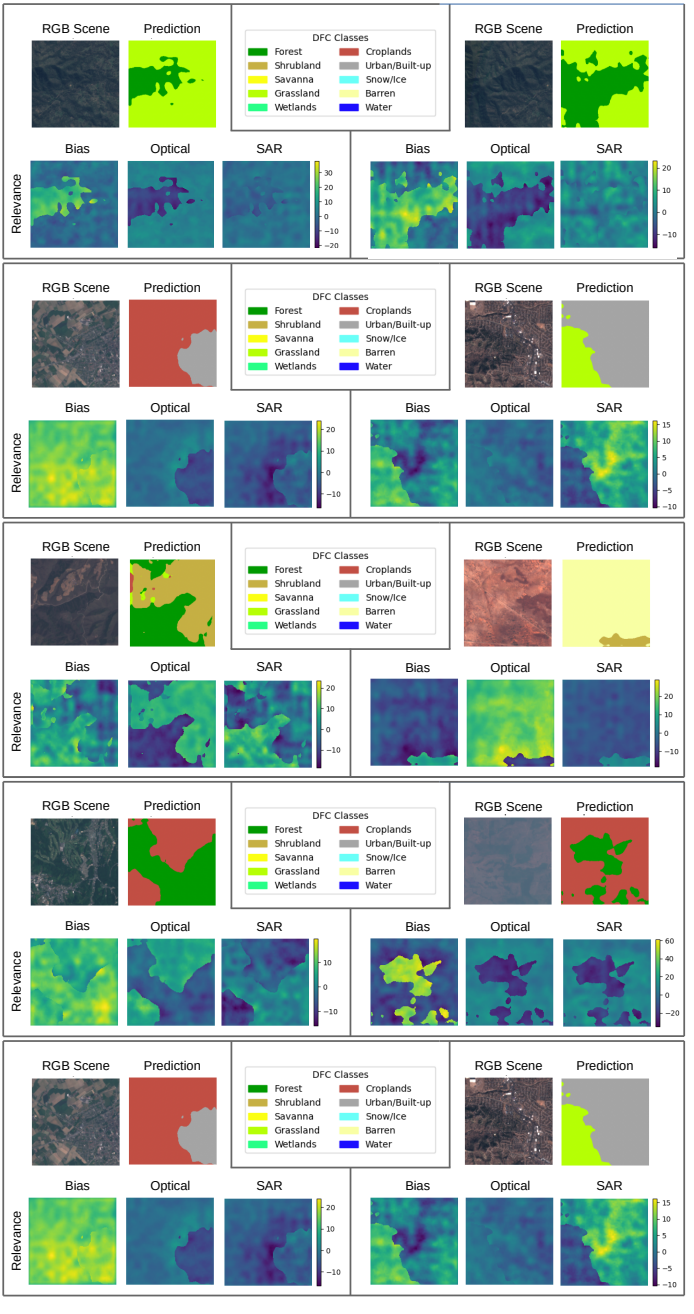

Figure S32: Examples of the relevance values based on the pre-trained DeepLabV3+ network of the DFC2020 baseline.

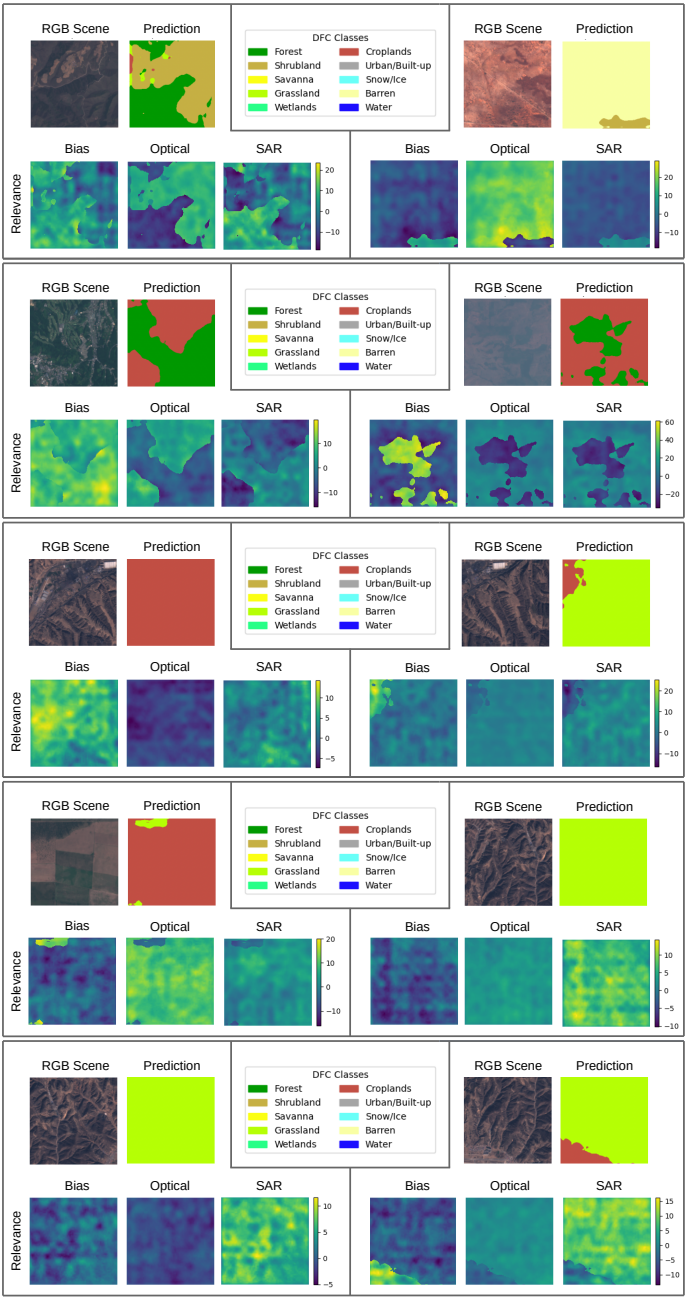

Figure S33: Examples of the relevance values based on the pre-trained DeepLabV3+ network of the DFC2020 baseline.

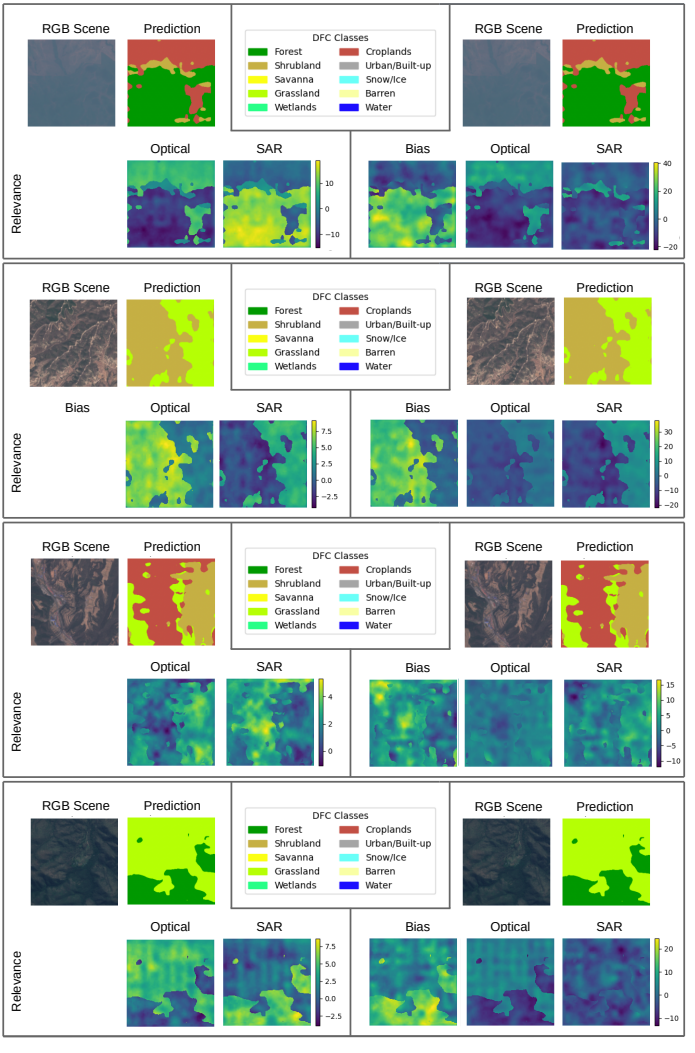

Figure S34: Examples of the relevance values based on the pre-trained DeepLabV3+ network of the DFC2020 baseline. The right side of each box shows the bias relevance incorporated into the optical and SAR relevance values as described above. The right side of each box shows the relevance values based on a bias, optical, and SAR data source.

5.10 RFP with not Piece-Wise Linear Functions

In this work, we consider two different types of how non-linearities can occur in neural networks.

Multiplication / Attention Mechanisms. The first type is based on multiplying two values in relevance representation. This is very common as the attention mechanisms in modern transformer architectures are designed like this. For this case, we propose to release the attention mechanism from the relevance representation, i.e., keep the attention a scalar weight instead of a vector of source-wise contributions to this weight. We applied this type of relevance linearization in the cloud removal example presented in Sec. 7.2.

Non-linear Operators. Non-linear operators such as sigmoid, tanh, elu, and others can be found. The following shows the multi-source MNIST examples with different non-linear activation functions for each layer. The relevance-weighted linearization and the Taylor linearization are based on the approaches described in Sec. 3.1.2 and Sec. 3.1.1.

For the ReLU, the ELU, and the tanh activation, the relevance values computed with RFP align very well with the relevance values of Integrated Gradients. However, the general trend can still be seen for the Sigmoid function, but the distributions look less clear with higher standard deviations between the sample values. We assume this is the case as it does not pass the origin such that the concept of positive and negative contribution does not propagate well through the network.

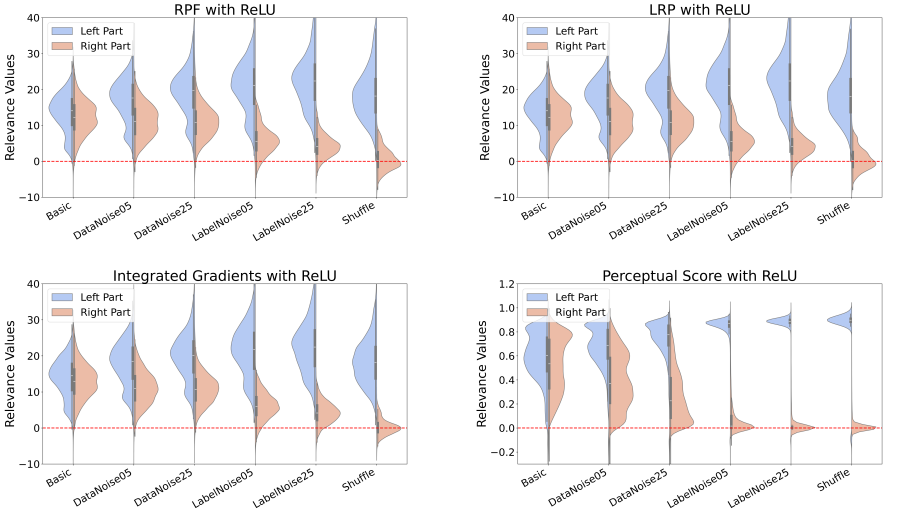

Figure S35: Basic Multi-Source MNIST with ReLU activations, evaluated with RFP, LRP, Integrated Gradients, and Perceptual Score.

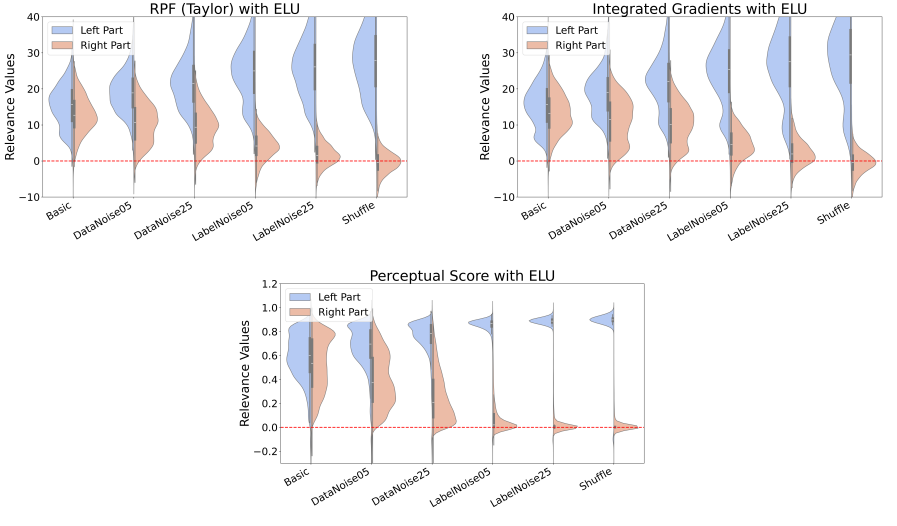

Figure S36: Basic Multi-Source MNIST with ELU activations, evaluated with RFP, LRP, Integrated Gradients, and Perceptual Score.

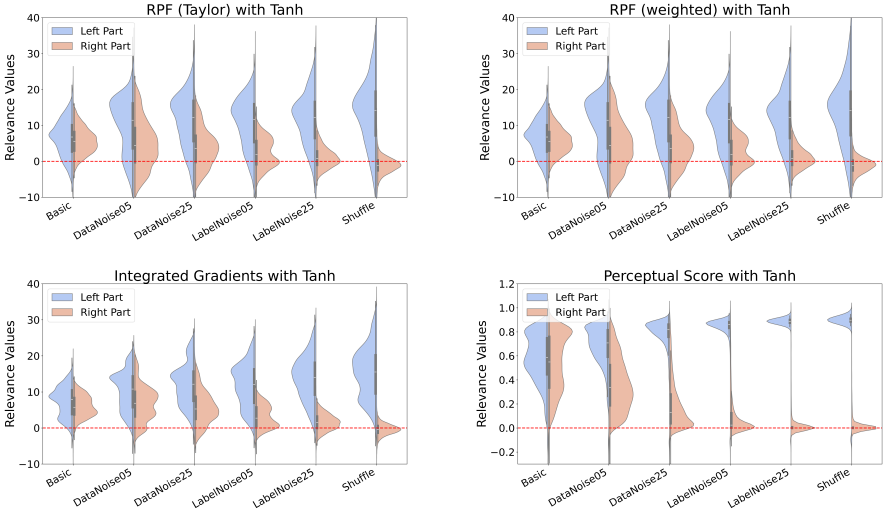

Figure S37: Basic Multi-Source MNIST with Tanh activations, evaluated with RFP, LRP, Integrated Gradients, and Perceptual Score.

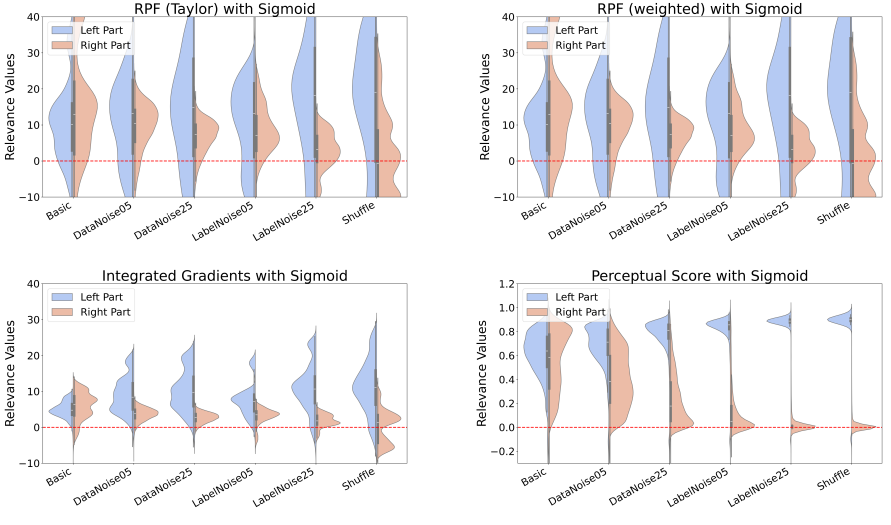

Figure S38: Basic Multi-Source MNIST with Sigmoid activations, evaluated with RFP, LRP, Integrated Gradients, and Perceptual Score.

6 Numerical Evaluation

6.1 Numerical Equivalence of Normal Forward Pass and Relevance Forward Propagation (RFP)

Even though the Forward Relevance Propagation does not change the prediction mathematically, numerical errors occur in every combination, also already for the original prediction. Hence, as both procedures are affected by different numerical errors, the results are not 100% equivalent. In the following table, we show how the Relevance Forward Propagation differs from the original network with respect to the prediction (L1-error), the prediction (accuracy), and loss. We show the results for data type float32 and float64/double. In order to set the

Table S4: L1 distance between a prediction computed with a normal neural network and the corresponding RFP version.

Dataset	Float32 Precision	Float64/Double Precision
MNIST Basic	$1.4908e^{-3} \pm 1.1820e^{-3}$	$2.8253e^{-15} \pm 2.6533e^{-15}$
SEN12MS Clear	$4.1753e^{-3} \pm 3.5791e^{-3}$	$8.5253e^{-15} \pm 7.2857e^{-15}$
SEN12MS Cloudy	$3.5376e^{-3} \pm 3.1238e^{-3}$	$7.2221e^{-15} \pm 6.3498e^{-15}$

results into relation, we also compute the L1-difference between the predictions of a normal neural network trained on Float32 and evaluated on Float32 and Float64 precision.

Table S5: L1-difference in the predictions when evaluating a pre-trained network with precision Float32 and precision Float64.

Dataset	L1-Distance
MNIST (Basic)	$1.5142e^{-3} \pm 1.2011e^{-3}$
SEN12MS Clear	$2.1992e^{-3} \pm 1.8836e^{-3}$
SEN12MS Cloudy	$2.2466e^{-3} \pm 1.8939e^{-3}$

6.2 Forward Relevance Propagation (FRP) vs. Layer-wise Relevance Propagation (LRP)

We numerically show the equivalence of RFP and LRP. For the LRP approach, we use the implementation provided with the Captum Library[4] [9] for model interpretability for models implemented in PyTorch. Captum does not return the relevance for the bias source, so we manually compute it by subtracting the received input relevance scores from the back-propagated model output. It is important to note that LRP delivered non-values when we tried to set the stability value ε to zero. Further, the LRP uses the forward propagation step, so numerical differences propagate through the different steps. Also, for SEN12MS, LRP delivered unreasonable high values in the range of $\pm 10^9$, which seem to be caused by numerical instabilities. Hence, only the relevance values for the MNIST examples are stated here.

Table S6: Sample-wise L1-error between the source-wise predicted relevance via Forward Relevance Propagation (FRP) and Layer-wise Relevance Propagation (LRP). The MNIST example was the only use case where LRP implemented in Captum delivered led to reasonable examples without most values in the range of -10^{10} or $+10^{10}$. Further, the backpropagation with LRP includes an ε value for numerical stability, which is not needed for RFP and which resulted in larger deviations for single precision computations.

Dataset	Float32 Precision	Float64/Double Precision
MNIST Basic	1.14797 ± 0.9200	$1.195 \cdot 10^{-7} \pm 5.4839 \cdot 10^{-6}$

6.3 Additional Computation Time Experiments on MNIST Data

The following table shows the computation time for predictions with RFP, LRP, and Perceptual Score on the Basic Multi-Source MNIST version. The experiments are run on an NVIDIA GeForce RTX 3070 GPU. The time is measured for the entire evaluation loop in which the relevance values are computed. The relevance is computed for one sample at a time for over all 10000 samples of the testing set. For LRP and Integrated Gradients, the Captum package is utilized with the default parameter setup[4].

Table S7: Copmutation times for different approaches on the Basic Multi-Source MNIST Version.

	Time [sec]	RFP Factor
RFP	15.46 ± 0.25	1
LRP	93.11 ± 1.05	6.02
Integrated Gradients	158.73 ± 2.62	10.42
Perc Score (5)	42.37 ± 0.45	2.74
Perc Score (10)	49.88 ± 0.50	3.23
Perc Score (15)	64.72 ± 0.66	4.18
Perc Score (25)	112.84 ± 2.42	7.30
Perc Score (50)	178.24 ± 2.33	11.53
Perc Score (100)	362.24 ± 3.54	23.43

[4] https://captum.ai/

7 Relevance Representation for Regression Problems

For a classification task, the prediction is determined by the largest logit. This means if a data source has a negative relevance for a logit related to a class, the class becomes less likely because of information extracted from the corresponding data source. This idea of relevance is not directly transferable to the regression setup, as regression tasks aim to predict a concrete value, and "negative" relevance might be needed if the bias is learned higher than the prediction or the actual value to predict is negative. As the bias also depends on the prediction and interplays with the relevance values from the individual data sources, a clear metric to interpret contributions becomes difficult. In the following, we present a dummy setup to underline that the general idea is transferable to regression setups and a cloud removal example, where optical (partially) cloud-covered images are reconstructed based on the optical and additional SAR information.

7.1 Regression Dummy Example

As a dummy example we consider the function $f(x_1,x_2) = x_1^3 + 0.1 \cdot \varepsilon$ with $\varepsilon \sim \mathcal{N}(0,1)$ with $x_1,x_2 \in \mathbb{R}$ and trained on four linear layers with first input dimension equal to 2, inner dimensions of 32 and output dimension of 1. We randomly sample x_1 and x_2 from the range $(-3,3)$ for the training. The results are shown in the plot below, and the pure relevance of data source 1 is visible due to the constant close-to-zero relevance related to modality 2. However, the prediction is not purely based on x_1 but also considers the bias values, which represent an inverse representation of the function to balance out the relevance curve of source 1.

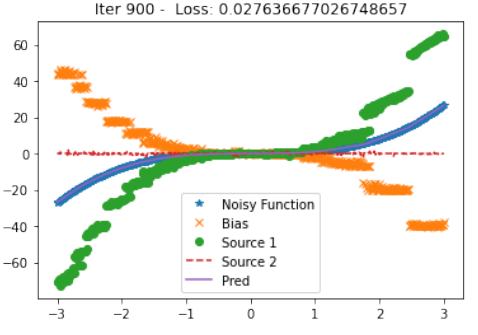

Figure S39: Evaluation of the source-wise relevance applied to the function $f(x_1,x_2) = x_1^3 + 0.1 \cdot \varepsilon$ with $\varepsilon \sim \mathcal{N}(0,1)$.

7.2 Regression Cloud Removal

For the following examples we used the repository of the UnCRtainTS network [2] (https://github.com/PatrickTUM/UnCRtainTS) and adjusted the pre-trained network without uncertainties and with only one time-step, equivalently as for the segmentation task above. The results here show a separation in the data source between cloudy and non-cloudy regions. However, it is not directly visible that the optical modality is used when no clouds cover the

input sample and not when clouds cover the optical ground truth. These results have multiple possible explanations: The network might not have been trained on less cloudy data and, hence, should not have been used to focus on the optical regions. Also, non-cloudy regions in the optical input can deliver information about color and general illumination setups that are not visible on the SAR data but can be transferred to cloud-covered regions. Further, non-cloudy regions could be covered by clouds, such that the optical data here might be less useful for detecting the underlying landcover type and structure than the SAR data.

Below, you find three ***explicitly selected examples*** for the cloud removal use-case. The question of the possibilities of quantifying or interpreting the contribution from individual data sources will be one of the focuses of future research. For the case with thin clouds, the contribution of the optical data sources is the highest; for the thick cloud coverage, the SAR modality is more relevant, and for the third example, the optical modality becomes more relevant for the parts that are not covered by the thick clouds.

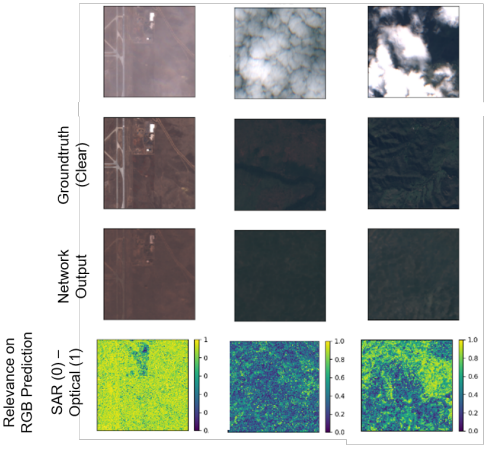

Figure S40: Three selected examples of the relevance of the SAR and optical modality on the cloud removal regression task. The relevance scale ranges from 0 (dark - total relevance on SAR) to 1 (bright - total relevance on optical).

Besides these three picked examples that deliver intuitively nice results, we also present the whole batch of images in Fig. S41.

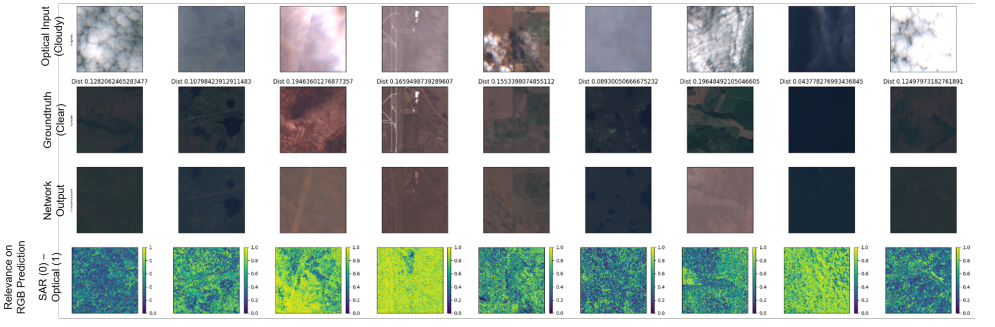

Figure S41: Multiple examples of the SAR and optical modality on the cloud removal regression task. The relevance scale ranges from 0 (dark - total relevance on SAR) to 1 (bright - total relevance on optical).

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[2] Patrick Ebel, Vivien Sainte Fare Garnot, Michael Schmitt, Jan Dirk Wegner, and Xiao Xiang Zhu. Uncrtaints: Uncertainty quantification for cloud removal in optical satellite time series. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2085–2095, 2023.

[3] Frederic Font Corbera, Gerard Roma Trepat, and Xavier Serra. Freesound technical demo. In *MM'13. Proceedings of the 21st ACM international conference on Multimedia; 2013 Oct 21-25; Barcelona, Spain. New York: ACM; 2013. p. 411-2.* ACM Association for Computer Machinery, 2013.

[4] Mark A Friedl, Douglas K McIver, John CF Hodges, Xiaoyang Y Zhang, D Muchoney, Alan H Strahler, Curtis E Woodcock, Sucharita Gopal, Annemarie Schneider, Amanda Cooper, et al. Global land cover mapping from modis: algorithms and early results. *Remote sensing of Environment*, 83(1-2):287–302, 2002.

[5] Itai Gat, Idan Schwartz, and Alex Schwing. Perceptual score: What data modalities does your model perceive? *Advances in Neural Information Processing Systems*, 34: 21630–21643, 2021.

[6] Jakob Gawlikowski, Patrick Ebel, Michael Schmitt, and Xiao Xiang Zhu. Explaining the effects of clouds on remote sensing scene classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:9976–9986, 2022.

[7] Di Hu, Xuhong Li, Lichao Mou, Pu Jin, Dong Chen, Liping Jing, Xiaoxiang Zhu, and Dejing Dou. Cross-task transfer for geotagged audiovisual aerial scene recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 68–84. Springer, 2020.

[8] Pengbo Hu, Xingyu Li, and Yi Zhou. SHAPE: An unified approach to evaluate the contribution and cooperation of individual modalities. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3064–3070. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/425. URL https://doi.org/10.24963/ijcai.2022/425. Main Track.

[9] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

[10] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: An overview. In Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 193–209. Springer International Publishing, Cham, 2019.

[11] M Schmitt and Y-L Wu. Remote sensing image classification with the sen12ms dataset. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 52:101–106, 2021.

[12] Michael Schmitt, Lloyd H Hughes, Chunping Qiu, and Xiao Xiang Zhu. Sen12ms–a curated dataset of georeferenced multi-spectral sentinel-1/2 imagery for deep learning and data fusion. *PIA19: Photogrammetric Image Analysis*, pages 153–160, 2019.