Interpretable Neural Networks with Frank-Wolfe: Sparse Relevance Maps and Relevance Orderings

Jan Macdonald [1] **Mathieu Besançon** [2] **Sebastian Pokutta** [1 2]

Abstract

We study the effects of constrained optimization formulations and Frank-Wolfe algorithms for obtaining interpretable neural network predictions. Reformulating the *Rate-Distortion Explanations (RDE)* method for relevance attribution as a constrained optimization problem provides precise control over the sparsity of relevance maps. This enables a novel multi-rate as well as a relevance-ordering variant of RDE that both empirically outperform standard RDE and other baseline methods in a well-established comparison test. We showcase several deterministic and stochastic variants of the Frank-Wolfe algorithm and their effectiveness for RDE.

1. Introduction

Deep learning methods achieve outstanding results for tasks across various fields, ranging from image analysis (Krizhevsky et al., 2012; Szegedy et al., 2013), to natural language processing (Cho et al., 2014; Vaswani et al., 2017), to medical diagnosis (Shen et al., 2017; McBee et al., 2018). However, they are mostly considered as black-box models. The reasoning of parameter-rich and highly nonlinear neural networks remains generally inaccessible. This is particularly undesirable in sensitive applications, such as medical diagnosis or autonomous driving.

The ability to render these models less opaque by providing human-interpretable explanations of their predictions is essential for a reliable use of neural networks. An important first step is the identification of the most relevant input features for a prediction, as illustrated in Figure 1.

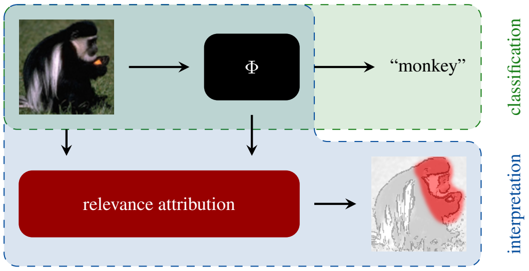

Figure 1. Relevance attribution methods aim at rendering black-box classifiers more interpretable by providing heatmaps of the input features that contribute most to an individual prediction.

This has recently been formalized as an optimization problem in a rate-distortion framework (Macdonald et al., 2019), cf. Figure 2. We give a brief informal introduction here. A more detailed description is given in Section 2.1. The rate-distortion approach aims at balancing the expected change in the model prediction when modifying the non-relevant input features (distortion D) against the number of features that are considered relevant (rate R). We propose to restate the original Lagrangian formulation

$$\min_{\mathbf{s}} \; D(\mathbf{s}) + \lambda \cdot R(\mathbf{s}), \tag{1}$$

with regularization parameter $\lambda > 0$ as a rate-constrained formulation

$$\min_{\mathbf{s} \in \mathcal{C}} \; D(\mathbf{s}), \tag{2}$$

with a feasible region of the form $\mathcal{C} = \{\mathbf{s} : R(\mathbf{s}) \leq k\}$ for some $k \in \mathbb{N}$. This has the advantage of providing finer control over the trade-off between the rate and the distortion, by precisely prescribing the number k of relevant features.

Variants of Gradient Descent (GD)[1] are by far the most popular optimization methods when working with neural networks and have previously been used to solve the Lagrangian formulation (1).[2] Incorporating the constraint $\mathbf{s} \in \mathcal{C}$ of (2) could also be achieved through Projected Gradient Descent (PGD). This requires a projection step

$$\mathbf{s}_{t+1} = \operatorname{proj}_{\mathcal{C}} \left(\mathbf{s}_t - \eta_t \nabla D(\mathbf{s}_t) \right)$$

[1] Institut für Mathematik, Technische Universität Berlin, Germany [2] Department for AI in Society, Science, and Technology, Zuse Institute Berlin, Germany. Correspondence to: Jan Macdonald <macdonald@math.tu-berlin.de>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

[1] A list of abbreviations can be found in Appendix A.

[2] More generally, proximal methods can be considered for non-smooth problems, e.g., including an ℓ_1-norm sparsity penalty as the rate function, cf. Section 2.1.

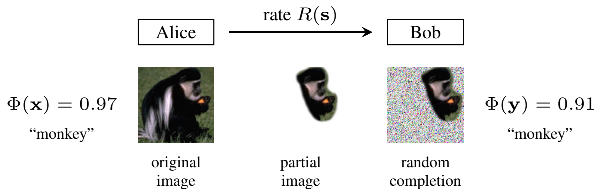

Figure 2. Illustration of the rate-distortion viewpoint of relevance attribution. In a hypothetical scenario, Alice and Bob are given access to a neural network classifier. Alice also has an image classified as a "monkey" and wants to convey this to Bob. He does not have the image and Alice is only allowed to send him a limited number of pixels. Bob will fill the remaining image with random values before classification. The best option is to transmit pixels that were most relevant for the prediction "monkey" of the original image. Figure adapted from Macdonald et al. (2020).

with step size $\eta_t > 0$ for each update in order to maintain feasible iterates.

Depending on the feasible region $\mathcal{C}$ such projections can be costly. An alternative projection-free first-order method is the Frank-Wolfe (FW) algorithm that relies on a (often computationally cheaper) Linear Minimization Oracle (LMO)

$$\mathbf{v}_t = \operatorname{argmin}_{\mathbf{v}\in\mathcal{C}} \langle \nabla D(\mathbf{s}_t), \mathbf{v} \rangle\,, \tag{3}$$

and then moves in direction $\mathbf{v}_t$ via the update

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \eta_t(\mathbf{v}_t - \mathbf{s}_t),$$

with step size $\eta_t \in [0, 1]$. Feasibility is maintained for convex regions $\mathcal{C}$ since the new iterate is a convex combination of two feasible points $\mathbf{s}_t$ and $\mathbf{v}_t$.

In this work, we examine the effectiveness of using FW (and variants thereof) for solving rate-constrained problems of the form (2) with the goal of obtaining interpretable neural networks.

Related Work A formal definition of the relevance of individual input features towards a prediction was proposed by Wäldchen et al. (2021) for classifiers on discrete domains and by Macdonald et al. (2019) for classifiers on continuous domains. It was shown that it is generally a hard computational problem to determine small sets of relevant features in the discrete setting (Wäldchen et al., 2021) and that this hardness carries over to the continuous setting (more specifically to neural networks) (Macdonald et al., 2020). We consider a problem-relaxation and heuristic solution strategy, named *Rate-Distortion Explanations* (RDE), introduced by Macdonald et al. (2019). This results in the Lagrangian formulation (1) and is described in more detail in Section 2.1.

Similarly, Fong & Vedaldi (2017) consider different types of perturbations of the non-relevant features. This also results in an optimization problem in the spirit of (1) that could be restated as a constrained problem. However, for brevity, we restrict our analysis of the efficacy of FW algorithms to the setting of RDE. Other explanation methods that are not based on optimization are discussed in Section 2.

It is worth mentioning, that the objective function in (2) will be non-convex. Although the Frank-Wolfe algorithm was originally intended for convex problems, it has also been studied and applied as a local method for problems with non-convex objectives over convex regions (Lacoste-Julien, 2016; Reddi et al., 2016). In particular, FW has recently received attention also in the context of neural networks, e.g., by Pokutta et al. (2020); Berrada et al. (2021).

Contributions This work extends and improves several aspects of the original RDE approach, which has been established as a promising and theoretically sound method for relevance attribution. We propose a rate-constrained formulation of RDE. Solving it with Frank-Wolfe algorithms yields sparse explanations and provides precise control over the size of the set of relevant features. This allows us to introduce a novel multi-rate variant of RDE. Further, in addition to sparsity constraints FW enables us to explore other feasible regions. We propose a novel method aiming at ordering input features according to their relevance for the prediction instead of a partition into relevant and irrelevant features. To this end, we optimize over the Birkhoff polytope (the convex relaxation of the set of permutation matrices). Our empirical study confirms the efficacy and flexibility of Frank-Wolfe algorithms for obtaining interpretable neural network predictions. We show for two exemplary image classification benchmark datasets that FW is able to determine relevant input features. Furthermore, the multi-rate as well as the relevance ordering variant of RDE both outperform standard RDE on a well-established comparison test.

Outline We review important concepts of interpretable neural networks in Section 2. A formal definition of RDE is given in Section 2.1. The idea of ordering input features according to their relevance is presented in Section 2.2. Section 3 and Appendix B provide details regarding variants of the Frank-Wolfe algorithms, feasible regions $\mathcal{C}$, and their linear minimization oracles. The description of our empirical study and its results are found in Section 4 and Appendix C.

2. Interpretable Neural Networks

In an abstract sense, *interpretability* refers to the ability to *explain* the predictions made by a deep learning model (or more generally: a machine learning model) to humans in an *understandable* way (Doshi-Velez & Kim, 2017). Here, "explain" and "understandable" are rather vague terms and can mean different things depending on the context and application. In an effort to address this task from any possible angle, there has been a surge of research related to explainable artificial intelligence (XAI) and various attempts to categorize interpretability methods, see e.g., the recent surveys by Fan et al. (2021); Chakraborty et al. (2017); Gilpin et al. (2018); Zhang et al. (2021).

Zhang et al. (2021) propose a taxonomy to distinguish methods according to three characteristics: i) *passive* methods give post hoc explanations for already trained neural networks while *active* methods require changing the network architectures already during training, ii) the *type of explanation* (in increasing order of explanatory power) ranges from extracting prototypical examples, to feature attribution, to extracting hidden semantics, and finally to extracting specific logical decision rules, iii) *local* methods explain network predictions for individual data samples while *global* methods aim at explaining networks as a whole.

We focus on passive and local feature attribution, which is among the most widely used forms of explanations. In this case, the goal is to assign scores to each input feature of a data sample indicating its *relevance* for a prediction, cf. Figure 1. Feature attribution methods are particularly popular in the context of image classification, where the scores are visualized as so-called *heatmaps* or *relevance maps*. Besides optimization based approaches such as RDE, various other relevance attribution methods for neural networks have been proposed, e.g., gradient based methods such as *Sensitivity Analysis* (Simonyan et al., 2013) and *SmoothGrad* (Smilkov et al., 2017), backward propagation methods such as *Guided Backprop* (Springenberg et al., 2015), *Layerwise Relevance Propagation* (LRP) (Bach et al., 2015), and *Deep Taylor Decompositions* (Montavon et al., 2018), surrogate model methods such as *Local Interpretable Model-Agnostic Explanations* (LIME) (Ribeiro et al., 2016), and game theoretic approaches such as *Shapley Additive Explanations* (SHAP) (Lundberg & Lee, 2017). We use these methods as comparison baselines in our numerical evaluations.

2.1. Sparse Relevance Maps

Intuitively, the relevant input features of a neural network classifier[3] $\Phi\colon \mathbb{R}^n \to \mathbb{R}$ for an input sample $\mathbf{x} \in \mathbb{R}^n$ are determined by the following desiderata: changing the non-relevant features of $\mathbf{x}$ while keeping its relevant features fixed should not result in a large change in the classifier prediction $\Phi(\mathbf{x})$. On the other hand, we do not want to mark all features as relevant but only those that necessarily have to be kept fixed to leave the prediction unchanged. Encoding the relevance scores in a vector $\mathbf{s} \in [0,1]^n$, where 0 means least relevant and 1 means most relevant, this can be formulated as a rate-distortion tradeoff with a distortion function

$$D(\mathbf{s}) = \mathbb{E}_{\mathbf{n}\sim\mathcal{V}}\left[\left(\Phi(\mathbf{x}) - \Phi(\mathbf{s}\odot\mathbf{x} + (\mathbf{1}-\mathbf{s})\odot\mathbf{n})\right)^2\right], \quad (4)$$

measuring the expected quadratic change in the classifier prediction when randomizing parts of $\mathbf{x}$ determined by $\mathbf{s}$, and a rate function $R(\mathbf{s}) = \|\mathbf{s}\|_1$, measuring the size[4] of $\mathbf{s}$ (Macdonald et al., 2019; 2020). Here, $\odot$ denotes the Hadamard product and $\mathcal{V}$ is some chosen probability distribution on $\mathbb{R}^n$ that is used to modify the features of $\mathbf{x}$ that are not fixed by $\mathbf{s}$. Typical choices for $\mathcal{V}$ include Gaussians (Fong & Vedaldi, 2017), uniform distributions (Samek et al., 2017), constant baseline signals (Fong & Vedaldi, 2017), or local distributions around $\mathbf{x}$ (Ribeiro et al., 2018). In this work, we will consider Gaussian distributions, since more data-adaptive choices of $\mathcal{V}$ can even be detrimental for uncovering the reasoning of a classifier (Janzing et al., 2020; Macdonald et al., 2020).

The goal is to find the smallest $\mathbf{s}$ (in terms of the rate) achieving a certain distortion or vice-versa the minimal distortion possible for a fixed size of $\mathbf{s}$. While the former results in a computationally challenging optimization problem with a non-convex constraint (distortion-constraint), the latter problem has a convex feasible region and it is—as we demonstrate in this work—computationally feasible to compute local minima.

The original Rate-Distortion Explanation (RDE) approach addresses the tradeoff between rate and distortion as a box-constrained optimization problem via a Lagrangian formulation

$$\begin{aligned} &\min_{\mathbf{s}} D(\mathbf{s}) + \lambda\|\mathbf{s}\|_1 \\ &\;\text{s.t.}\;\; \mathbf{s} \in [0,1]^n, \end{aligned} \tag{L-RDE}$$

with a regularization parameter $\lambda > 0$. Instead, we propose to use the rate-constrained formulation

$$\begin{aligned} &\min_{\mathbf{s}} D(\mathbf{s}) \\ &\;\text{s.t.}\;\; \|\mathbf{s}\|_1 \leq k, \\ &\qquad \mathbf{s} \in [0,1]^n, \end{aligned} \tag{RC-RDE}$$

with a maximal rate $k \in [n-1] = \{1, \ldots, n-1\}$.[5] This allows for a precise control of the size of $\mathbf{s}$ or in other words for the *sparsity* of the relevance map. Controlling the sparsity makes it easier to exactly pinpoint the most relevant input features of $\mathbf{x}$, which would otherwise require more manual tuning of the regularization parameter λ.

[3] In multi-class problems the classifier would usually be a map $\widetilde{\Phi}\colon \mathbb{R}^n \to \mathbb{R}^{\#\text{classes}}$ instead of $\Phi\colon \mathbb{R}^n \to \mathbb{R}$. However, as common for local explanations, we only consider the restriction of $\widetilde{\Phi}$ to the component corresponding to the class predicted for the data sample $\mathbf{x}$.

[4] The ℓ_1-norm can be seen as a convex surrogate for the cardinality of the support of $\mathbf{s}$, i.e., the size of the *relevant set* of input components (those with positive relevance scores).

[5] The case $k = n$ always has the trivial solution $\mathbf{s} = \mathbf{1}$.

Computing the expectation value in (4) exactly is generally not possible. As in the original RDE approach, we exploit the layered structure of neural networks and rely on an approximation scheme based on *assumed density filtering* (Minka, 2001) for the evaluation of the non-convex function $D(\mathbf{s})$. The Gaussian distribution $\mathcal{V}$ is estimated from the training data and the gradient $\nabla D(\mathbf{s})$ is obtained through automatic differentiation. Using this first-order information, we can then solve (RC-RDE) with (variants of) the Frank-Wolfe algorithm (Frank & Wolfe, 1956) or PGD, see Section 3.

2.2. Relevance Orderings

The exact values of a relevance map are often not individually meaningful (in the sense that knowing the value s_i for the i-th variable alone is not helpful). It is rather the ordering (by relevance) of the variables induced by $\mathbf{s}$ that is of interest, i.e., the relations $s_i < s_j$ or $s_j < s_i$ between different variables. In fact, an established evaluation method for the comparison of different relevance maps, the *pixel-flipping* test (Samek et al., 2017), and a variant thereof proposed by Macdonald et al. (2019) is based on this idea. It proceeds as follows: i) order variables by their relevance ii) keep increasingly large parts of the input $\mathbf{x}$ fixed and randomize the remaining variables iii) observe the change in the classifier prediction (distortion) during this process. A good ordering will lead to a quick decrease of the distortion as truly important input features are fixed first. This precisely corresponds to the rate-distortion tradeoff described above. In fact, (L-RDE) and (RC-RDE) aim at optimizing the resulting rate-distortion curve for a single rate determined implicitly by λ or explicitly by k respectively, cf. Figure 5.

Instead of first obtaining relevance scores and afterwards retrieving a relevance ordering from them, one could find an optimal ordering directly by solving

$$\min_{\mathbf{\Pi}} \frac{1}{n-1} \sum_{k\in[n-1]} D(\mathbf{\Pi}\mathbf{p}_k)$$
$$\text{s.t. } \mathbf{\Pi} \in S_n,$$

where S_n denotes the set of $(n \times n)$ permutation matrices and $\mathbf{p}_k = \sum_{j=1}^{k} \mathbf{e}_j$ is the vector of k-ones and $(n-k)$-zeros. Hence, $D(\mathbf{\Pi}\mathbf{p}_k)$ corresponds to the distortion of fixing the k most relevant features (according to $\mathbf{\Pi}$) and the objective aims at minimizing the average distortion across all rates $k \in [n-1]$ simultaneously. We relax this combinatorial problem (S_n is discrete) to

$$\min_{\mathbf{\Pi}} \frac{1}{n-1} \sum_{k\in[n-1]} D(\mathbf{\Pi}\mathbf{p}_k) \qquad \text{(Ord-RDE)}$$
$$\text{s.t. } \mathbf{\Pi} \in B_n,$$

where S_n is replaced with its convex hull, i.e., the Birkhoff polytope $B_n = \operatorname{conv}(S_n)$ of doubly stochastic $(n \times n)$-matrices. This can be solved with a (batched) stochastic version of the Frank-Wolfe algorithm (Hazan & Luo, 2016) or as before with non-stochastic versions of Frank-Wolfe if n is small enough so that evaluating the complete sum in (Ord-RDE) is not too expensive. There is no exact projection method specific to the Birkhoff polytope, see Appendix B, hence we do not consider PGD for solving (Ord-RDE).

A solution to (Ord-RDE) can be seen as a greedy-approximation to (RC-RDE) across all rates $k \in [n-1]$ simultaneously in the following sense: a solution $\mathbf{\Pi}^{\text{opt}}$ is a convex combination of permutation matrices (the vertices of B_n). It can be used to obtain mappings for specific rates via $\mathbf{\Pi}^{\text{opt}}\mathbf{p}_k$, which we interpret as a convex combination of the respective k most relevant components according to each permutation contributing to a convex decomposition of $\mathbf{\Pi}^{\text{opt}}$. From now on we refer to $\mathbf{\Pi}^{\text{opt}}\mathbf{p}_k$ with $k \in [n-1]$ as the single-rate mappings associated to $\mathbf{\Pi}^{\text{opt}}$.

One should note that, in contrast to (RC-RDE), a straightforward approach to solving (Ord-RDE) is not feasible for large-scale problems: optimizing over matrices in $\mathbb{R}^{n\times n}$ instead of vectors in $\mathbb{R}^n$ results in increased computational costs, both in terms of memory requirements and runtime (see also descriptions of the LMOs in Appendix B). However, this might in part be remedied by a clever and more memory-efficient representation of iterates.[6]

Another possibility to overcome this limitation is to emulate a similar multi-rate strategy that relies solely on the rate-constrained formulation: we can separately solve (RC-RDE) for all $k \in \mathcal{K}$ for some $\mathcal{K} \subseteq [n-1]$ and combine these solutions, e.g. by averaging, to obtain relevance scores

$$\mathbf{s} = \frac{1}{|\mathcal{K}|} \sum_{k\in\mathcal{K}} \left\{ \begin{array}{ll} \operatorname{argmin} D(\mathbf{s}) & \\ \text{s.t.} & \|\mathbf{s}\|_1 \leq k, \\ & \mathbf{s} \in [0,1]^n, \end{array} \right\} \qquad \text{(MR-RDE)}$$

that take multiple rates into account. Consequently, this effectively yields an induced ordering of the variables according to their relevance across all rates (if a variable is contained in the solutions for many of the rates then it con-

[6] The t-th FW iterate $\mathbf{\Pi}_t$ is a convex combination of an active set of at most t permutations (corresponding to vertices of B_n). Storing these together with the convex weights allows to effectively recover the iterate but reduces the memory requirement from $\mathcal{O}(n^2)$ to $\mathcal{O}(tn)$ (assuming $t < n$). For algorithms keeping track of an active set anyways, such as Away-Step Frank-Wolfe, see Appendix B, there is no computational overhead. Similarly, a sparse matrix representation reduces the memory requirement to $\mathcal{O}(\#\text{non-zero components})$. However, the bottleneck of our current implementation is the LMO evaluation. The gradients ($\mathbf{G}_t$ in Algorithm 2) will still be dense matrices. In fact, they are sums of rank-1 matrices. We leave a study of the practical benefits of exploiting this structure for the LMO evaluation to future research.

tributes more to the averaging of relevance maps over $\mathcal{K}$ and should be considered more relevant than a variable that is only contained in few solutions, hence it should come earlier in the ordering).

In summary, we can interpret the different RDE variants in terms of their objective regarding the pixel-flipping evaluation: solving a single (RC-RDE) problem aims at optimizing the rate-distortion curve, i.e., achieving low distortion, at a single rate. This might lead to suboptimal results at other rates, see Figure 5. In contrast, (Ord-RDE) aims at directly optimizing the relevance ordering and thus optimizes the distortion for all rates simultaneously on average. Finally, (MR-RDE) combines single-rate solutions spread across the range of considered rates and thus approximately also aims at optimizing the distortion everywhere by minimization at well-chosen attachment points. This is only possible because the Frank-Wolfe methods allow us to efficiently solve the rate-constrained problem with precise control over the sparsity of the computed relevance maps.

3. Frank-Wolfe Algorithms

The Frank-Wolfe algorithm (Frank & Wolfe, 1956) or conditional gradient method (Levitin & Polyak, 1966) is a projection-free first-order algorithm for constrained optimization over a convex compact feasible set. Since its first appearance, several algorithmic variations have been developed that enhance the performance of the original algorithm, while maintaining many of its advantages. In our experiments, we consider vanilla Frank-Wolfe (FW), Away-Step Frank-Wolfe (AFW), Lazified Conditional Gradients (LCG), Lazified Away-Step Frank-Wolfe (LAFW), Stochastic Frank-Wolfe (SFW). For comparison we also consider Projected Gradient Descent (PGD). We give a brief overview of FW and SFW here and defer a description of AFW, LCG, and LAFW as well as of the feasible regions $\mathcal{C}$ and associated LMOs and projections to Appendix B. The interested reader is referred to a more detailed presentation of the algorithm variants and their implementations[7] (Besançon et al., 2021).

Vanilla Frank-Wolfe (FW) In its basic version, see Algorithm 1, a linear approximation to the objective function at the current iterate (obtained from first-order information) is minimized over the feasible region. Such a linear minimization oracle (LMO) is often computationally less costly than a corresponding projection (which amounts to solving a quadratic problem). The next FW iterate is obtained as a convex combination of the solution to the LMO and the current iterate. For convex regions $\mathcal{C}$, this guarantees that the algorithm produces iterates that remain feasible throughout all iterations. This basic variant has the lowest memory requirements among all deterministic variants since it only requires keeping track of the current iterate. Hence, it is well suited for large-scale problems. However, other variants can achieve improvements in terms of convergence speed (iteration count and time for more specialized setups), see Appendix B.

Algorithm 1 Frank-Wolfe for (RC-RDE)

```
Input: initial guess $\mathbf{s}_0 \in \mathcal{C} = \{\mathbf{s} \in [0,1]^n : \|\mathbf{s}\|_1 \leq k\}$,
    number of steps $T$, step sizes $\eta_t \in [0,1]$
Output: a stationary point $\mathbf{s}^{\text{opt}}$ of (RC-RDE)
for $t \leftarrow 1$ to $T$ do
    $\mathbf{v}_t \leftarrow \operatorname{argmin}_{\mathbf{v} \in \mathcal{C}} \langle \nabla D(\mathbf{s}_{t-1}), \mathbf{v} \rangle$
    $\mathbf{s}_t \leftarrow \mathbf{s}_{t-1} + \eta_t(\mathbf{v}_t - \mathbf{s}_{t-1})$
end for
return $\mathbf{s}_T$
```

Stochastic Frank-Wolfe (SFW) In some cases, evaluating the full objective function and its gradients is expensive, but cheaper unbiased estimators are available. The typical example is that of objective functions that are sums of a large number of terms, such as in the (Ord-RDE) formulation. An estimator is given by evaluating the sum only over a randomly chosen subset of terms. The stochastic version of Frank-Wolfe (Hazan & Luo, 2016) developed in Algorithm 2 uses a gradient estimate instead of the exact gradient in combination with a momentum term (Mokhtari et al., 2020) to build the linear approximation to the objective in each iteration. Then the LMO is evaluated and a step is taken exactly as in the vanilla FW algorithm. Different variants of SFW have recently been studied also in the non-convex setting (Yurtsever et al., 2019; Shen et al., 2019; Hassani et al., 2020; Négiar et al., 2020).

Algorithm 2 Stochastic Frank-Wolfe for (Ord-RDE)

```
Input: initial guess $\mathbf{\Pi}_0 \in B_n$, number of steps $T$, step
    sizes $\eta_t \in [0,1]$, batch sizes $b_t \in [n-1]$, momentum
    factors $\rho_t \in [0,1]$
Output: a stationary point $\mathbf{\Pi}^{\text{opt}}$ of (Ord-RDE)
$\mathbf{M}_0 \leftarrow \mathbf{0}_{n \times n}$
for $t \leftarrow 1$ to $T$ do
    sample $k_1, \ldots, k_{b_t}$ i.i.d. uniformly from $[n-1]$
    $\mathbf{G}_t \leftarrow \frac{1}{b_t} \sum_{j=1}^{b_t} \nabla D(\mathbf{\Pi}_{t-1}\mathbf{p}_{k_j})\mathbf{p}_{k_j}^\top$
    $\mathbf{M}_t \leftarrow \rho_t \mathbf{M}_{t-1} + (1-\rho_t)\mathbf{G}_t$
    $\mathbf{V}_t \leftarrow \operatorname{argmin}_{\mathbf{V} \in B_n} \langle \mathbf{M}_t, \mathbf{V} \rangle$
    $\mathbf{\Pi}_t \leftarrow \mathbf{\Pi}_{t-1} + \eta_t(\mathbf{V}_t - \mathbf{\Pi}_{t-1})$
end for
return $\mathbf{\Pi}_T$
```

[7] We use the `FrankWolfe.jl` Julia package available at https://github.com/ZIB-IOL/FrankWolfe.jl for our experiments. See Appendix B for more details regarding our computational setup.

Parameter Choices The basic step size rule

$$\eta_t = \frac{1}{\sqrt{t+1}} \tag{5}$$

can be used for non-convex objectives (Reddi et al., 2016; Combettes, 2021). An adaptive step-size choice similar to one proposed by Carderera et al. (2021) is

$$\eta_t = \frac{2^{-r_t}}{\sqrt{t+1}} \tag{6}$$

where $r_t \in \mathbb{N}$ is found by repeated increments starting from r_{t-1} until primal progress is made. This ensures monotonicity in the objective, which is not necessarily the case for the basic rule (5). In our experiments, we use the monotone rule (6) for FW, AFW, LCG, and LAFW and a corresponding (rescaled) variant also for the PGD comparison. Enforcing monotonicity does not make sense in the stochastic setting and we use the basic rule (5) for SFW. We test multiple configurations of SFW with constant or increasing batch sizes and momentum factors as proposed by Hazan & Luo (2016) and Mokhtari et al. (2020) respectively. The full results can be found in Appendix C. In the next section, we only show the best performing configuration with constant batch size $b_t = 40$ and no momentum, i.e., $\rho_t = 0$. In all cases, we terminate after a maximal number of $T = 2000$ iterations or if the dual gap $\langle \mathbf{s}_t - \mathbf{v}_t, \nabla D(\mathbf{s}_t)\rangle$ (respectively $\langle \mathbf{\Pi}_t - \mathbf{V}_t, \mathbf{M}_t\rangle$ for SFW) drops below the prescribed threshold $\varepsilon = 10^{-7}$.

4. Computational Results

We generate relevance mappings for neural network classifiers trained on two image classification benchmark tasks. The first consists of greyscale images of handwritten digits from the MNIST dataset (LeCun et al., 1998) and the second consists of color images from the STL-10 dataset (Coates et al., 2011). We use the relevance ordering-based comparison test, as described in Section 2.2, for a quantitative evaluation of the relevance maps in addition to a purely visual qualitative evaluation. We show results for (RC-RDE), (MR-RDE), and (Ord-RDE) as well as (L-RDE) and several established relevance mapping methods as comparison baselines.[8]

MNIST Experiment For direct comparability, we use the same convolutional neural network of Macdonald et al. (2019) (with three convolutional layers each followed by average-pooling and finally two fully-connected layers and softmax output) that was trained end-to-end up to test accuracy of 0.99. The relevance mappings are calculated for the pre-softmax score of the class with the highest activation.

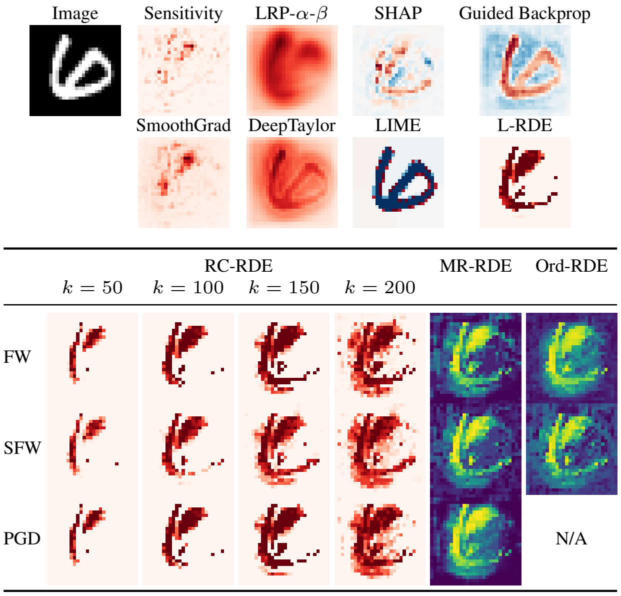

Figure 3. Relevance mappings for an MNIST image classified as *digit six* by the network. The colormap indicates positive relevance as red and negative relevance as blue. Multi-rate solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but as component orderings from least relevant (blue) to most relevant (yellow). Results for the other FW variants are shown in Figure 9 in the appendix.

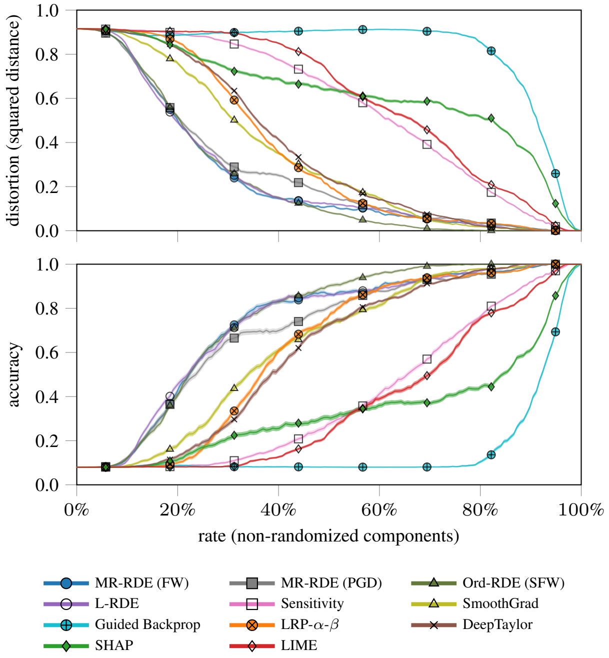

Figure 4. Relevance ordering test results for MNIST for all considered methods. An average result over 50 images from the test set (5 images per class) and 512 noise input samples per image is shown (shaded regions mark ± standard deviation). A comparison of different FW variants for the RDE method is shown in Figure 11 in the appendix.

[8] Our code is available at `https://github.com/ZIB-IOL/fw-rde` (Macdonald et al., 2021). All (L-RDE) and comparison results were provided by Macdonald et al. (2019). Their code is available at `https://github.com/jmaces/rde`. Some methods give relevance scores in $[-1, 1]$ instead of $[0, 1]$ (indicating contributions *against* and *towards* the prediction).

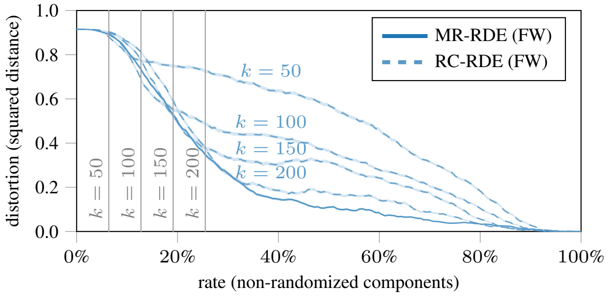

Figure 5. Relevance ordering test results for MNIST and (RC-RDE) at various rates. Vertical lines show the rates k at which the mappings were optimized. The combined (MR-RDE) solution approximates a lower envelope of the individual curves. An average result over 50 images from the test set (5 images per class) and 512 noise input samples per image is shown (shaded regions mark ± standard deviation).

The mappings for an example image of a digit six are shown in Figure 3. It shows (RC-RDE) mappings for FW and PGD at different selected rates k, as well as respective (MR-RDE) mappings with $\mathcal{K} = \{50, 100, 150, 200, 250, 300, 350, 400\}$ and an (Ord-RDE) mapping.[9] The middle row shows the corresponding single-rate mappings $\mathbf{\Pi}^{\mathrm{opt}}\mathbf{p}_k$ associated to the SFW solution of (Ord-RDE) as well as a corresponding multi-rate mapping $\frac{1}{n-1}\sum_{k\in[n-1]}\mathbf{\Pi}^{\mathrm{opt}}\mathbf{p}_k$. Mappings for the other FW variants are shown in Figure 9 in the appendix. All RDE variants generate similar results and highlight an area at the top as relevant, that distinguishes a six from the digits zero and eight. All FW methods and PGD are robust across varying rates, in the sense that solutions for larger rates add additional features to the relevant set without significantly modifying the features that were already considered relevant at smaller rates. The (L-RDE) solution is most similar to the (RC-RDE) solutions at rate $k = 100$.

The quantitative effect of solving (RC-RDE) for different rates is illustrated in Figure 5. For the sake of clarity, we only show the relevance ordering test results for FW. The results for AFW, LCG, LAFW, and PGD are comparable. The (RC-RDE) mappings achieve a low distortion at the rates for which they were optimized but are suboptimal at other rates. The combined (MR-RDE) solution approximates the lower envelope of the individual curves and performs best overall.

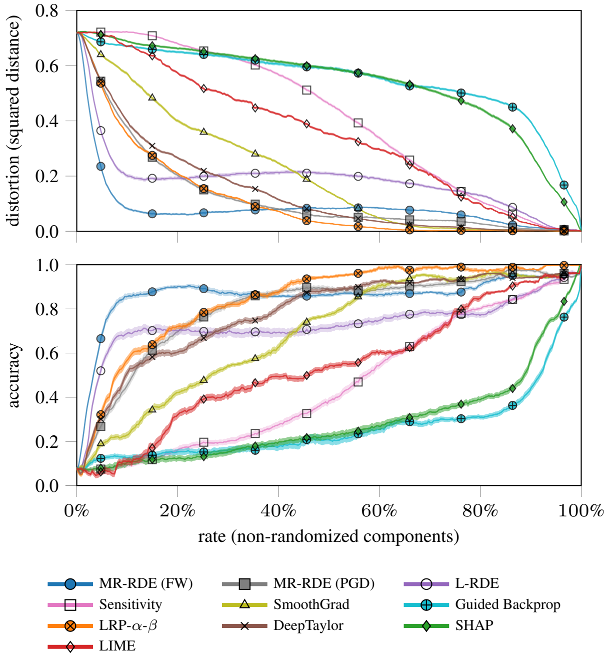

Figure 6. Relevance ordering test results for STL-10. An average result over 50 images from the test set (5 images per class) and 64 noise input samples per image is shown (shaded regions mark ± standard deviation). A comparison of different FW variants for the RDE method is shown in Figure 12.

Figure 4 shows a comparison of the relevance ordering test results for two different performance measures (distortion at the top, classification accuracy at the bottom). Figure 11 in the appendix shows the corresponding results for the other FW variants. All RDE methods result in a fast drop in the distortion (respectively a fast rise in the accuracy), indicating that the relevant features were correctly identified. They clearly outperform the other relevance attribution methods. The FW solutions perform slightly better than the PGD solutions. As expected, the (Ord-RDE) solution performs best overall.

STL-10 Experiment We consider the same VGG-16 network (Simonyan & Zisserman, 2014) pretrained on the Imagenet dataset and refined on STL-10 to a final test accuracy of 0.935 as used by Macdonald et al. (2019). The relevance mappings are again calculated for the pre-softmax score of the class with the highest activation.

The mappings for two example images of a monkey and a horse are shown in Figures 7 and 8. They show (RC-RDE) mappings for FW and PGD at different selected rates k, as well as respective (MR-RDE) mappings with $\mathcal{K} = \{2000, 4000, 6000, 8000, 10000, 14000, 18000, 22000, 26000, 30000, 34000, 38000, 42000, 46000, 50000\}$.[10] Corresponding mappings for the other FW variants are shown in Figure 15 in the appendix. All RDE methods generate similar results and highlight parts of the face and body of

[9] Images are 28×28 greyscale, hence $n = 28 \cdot 28 = 784$.

[10] Images are resized to 224×224 with three color channels, hence $n = 3 \cdot 224 \cdot 224 = 150528$. Mappings are visualized as a single channel heatmap that averages relevance scores across color channels.

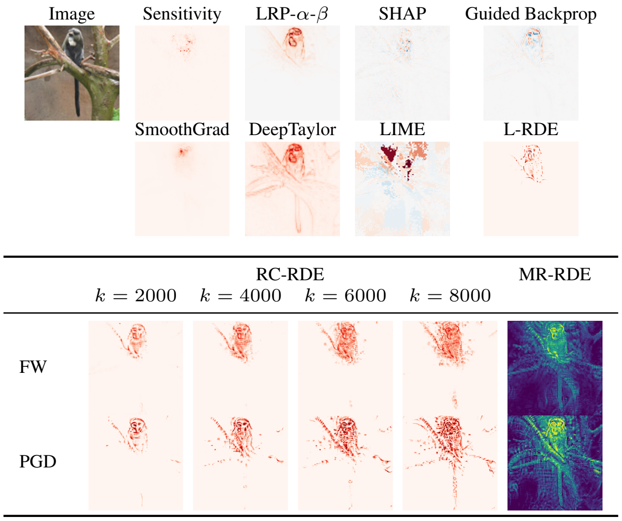

Figure 7. Relevance mappings for an STL-10 image classified as *monkey* by the network. The colormap indicates positive relevance as red and negative relevance as blue. Multi-rate solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but as component orderings from least relevant (blue) to most relevant (yellow). Results for the other FW variants are shown in Figure 15 (left) in the appendix.

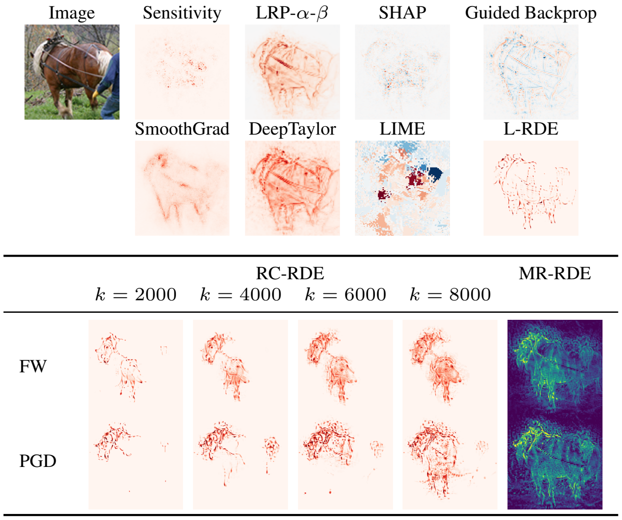

Figure 8. Relevance mappings for an STL-10 image classified as *horse* by the network. The colormap indicates positive relevance as red and negative relevance as blue. Multi-rate solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but as component orderings from least relevant (blue) to most relevant (yellow). Results for the other FW variants are shown in Figure 15 (right) in the appendix.

the monkey as relevant at small rates. Increasingly large parts of the body and tail of the monkey are added to the relevant set at higher rates. Similarly, parts of the head and front legs of the horse are marked relevant first and larger parts of its body get added at higher rates. As before, the results are robust across varying rates. Additional images are shown in Appendix D. Compared to MNIST, the difference between the FW and PGD solutions and between the Lagrangian and the rate-constrained formulations are more pronounced for STL-10. The sparsity of the (L-RDE) solution is most comparable to the (RC-RDE) solutions at rate $k = 2000$. However, it is less concentrated at a specific part of the body of the animals, especially for the horse image.

Figure 6 shows the relevance ordering test results for two different performance measures (distortion at the top, classification accuracy at the bottom). Figure 12 in the appendix shows the corresponding results for the other FW variants. Due to the large size of problem instances, we do not show (Ord-RDE) solutions in this experiment. All RDE methods result in a fast drop in the distortion (respectively a fast rise in the accuracy), indicating that the relevant features were correctly identified. Again, they clearly outperform the other relevance attribution methods, especially at the lower rates. The non-monotone behavior of the curves for rates between 30% and 70% can be explained by the fact that the maximal considered rate in $\mathcal{K}$ corresponds to about 33% of the total number of components. Unlike for MNIST, there is a considerable difference between the FW and PGD solutions for RDE and between (L-RDE) and (MR-RDE). The latter outperforms the original RDE version across all rates. FW outperforms PGD for rates up to 40%. Results are comparable for higher rates.

5. Discussion

We have proposed and analyzed a rate-constrained and a relevance-ordering variation of Rate-Distortion Explanations (RDE) for interpretable neural networks. Solutions can be efficiently obtained using Frank-Wolfe algorithms. While optimization based relevance attribution methods, such as RDE, are computationally more demanding than single-pass methods, such as, e.g., LRP and Guided Backprop, we believe that this does not hinder their practical use. On the contrary, meaningful interpretations and excellent performance in quantitative evaluations are of more interest in critical applications than mere runtimes. Further, we observe that already very few Franke-Wolfe iterations are sufficient for obtaining accurate RDE mappings, cf. Figure 10 in the appendix. These can be computed within seconds, which renders RDE feasible for all applications that do not require real-time interpretations.

Acknowledgements

Research reported in this paper was partially supported through the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund numbers 05M14ZAM, 05M20ZBM) and the Deutsche Forschungsgemeinschaft (DFG) through the DFG Cluster of Excellence MATH+.

References

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):1–46, 7 2015. doi: 10.1371/journal.pone.0130140.

Berrada, L., Zisserman, A., and Kumar, M. P. Deep Frank-Wolfe For Neural Network Optimization. Preprint, arXiv:1811.07591, 2021.

Besançon, M., Carderera, A., and Pokutta, S. FrankWolfe.jl: a high-performance and flexible toolbox for Frank-Wolfe algorithms and Conditional Gradients. Preprint, arXiv:2104.06675, 2021.

Braun, G., Pokutta, S., and Zink, D. Lazifying Conditional Gradient Algorithms. *J. Mach. Learn. Res.*, 20(1): 2577–2618, 1 2019.

Carderera, A., Besançon, M., and Pokutta, S. Simple steps are all you need: Frank-Wolfe and generalized self-concordant functions. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=rq_UD6IiBpX.

Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., Srivastava, M., Preece, A., Julier, S., Rao, R. M., Kelley, T. D., Braines, D., Sensoy, M., Willis, C. J., and Gurram, P. Interpretability of deep learning models: A survey of results. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld / SCALCOM / UIC / ATC / CBDCom / IOP / SCI)*, pp. 1–6, 2017. doi: 10.1109/UIC-ATC.2017.8397411.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, 10 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179.

Coates, A., Ng, A., and Lee, H. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In Gordon, G., Dunson, D., and Dudík, M. (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 4 2011. PMLR. URL http://proceedings.mlr.press/v15/coates11a.html.

Combettes, C. W. *Frank-Wolfe Methods For Optimization and Machine Learning*. PhD thesis, Georgia Institute of Technology, 2021.

Combettes, C. W. and Pokutta, S. Complexity of Linear Minimization and Projection on Some Sets. Preprint, arXiv:2101.10040, 2021.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. ISBN 978-0-262-03384-8. URL http://mitpress.mit.edu/books/introduction-algorithms.

Doshi-Velez, F. and Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. Preprint, arXiv:1702.08608, 2017.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the 25 th International Conference on Machine Learning*, 2008.

Fan, F.-L., Xiong, J., Li, M., and Wang, G. On Interpretability of Artificial Neural Networks: A Survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, pp. 1–1, 2021. doi: 10.1109/TRPMS.2021.3066428.

Fong, R. C. and Vedaldi, A. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, 2017.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. doi: https://doi.org/10.1002/nav.3800030109.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. Explaining Explanations: An Overview of Interpretability of Machine Learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 80–89, 2018. doi: 10.1109/DSAA.2018.00018.

Guélat, J. and Marcotte, P. Some Comments of Wolfe's 'away Step'. *Math. Program.*, 35(1):110–119, 5 1986.

Gupta, M. D., Kumar, S., and Xiao, J. L1 Projections with Box Constraints. Preprint arXiv:1010.0141, 2010.

Hassani, H., Karbasi, A., Mokhtari, A., and Shen, Z. Stochastic Conditional Gradient++. Preprint, arXiv:1902.06992, 2020.

Hazan, E. and Luo, H. Variance-Reduced and Projection-Free Stochastic Optimization. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1263–1271, New York, New York, USA, 06 2016. PMLR. URL http://proceedings.mlr.press/v48/hazana16.html.

Janzing, D., Minorics, L., and Bloebaum, P. Feature relevance quantification in explainable AI: A causal problem. In Chiappa, S. and Calandra, R. (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2907–2916. PMLR, 8 2020. URL http://proceedings.mlr.press/v108/janzing20a.html.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 25, pp. 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109.

Lacoste-Julien, S. Convergence Rate of Frank-Wolfe for Non-Convex Objectives. Preprint, arXiv:1607.00345, 2016.

Lacoste-Julien, S. and Jaggi, M. On the Global Linear Convergence of Frank-Wolfe Optimization Variants. In *NIPS 2015 - Advances in Neural Information Processing Systems 28*, Montreal, Canada, 12 2015.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 11 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Levitin, E. and Polyak, B. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966. doi: https://doi.org/10.1016/0041-5553(66)90114-5.

Lundberg, S. M. and Lee, S.-I. A Unified Approach to Interpreting Model Predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017.

Macdonald, J., Wäldchen, S., Hauch, S., and Kutyniok, G. A Rate-Distortion Framework for Explaining Neural Network Decisions. Preprint, arXiv:1905.11092, 2019.

Macdonald, J., Wäldchen, S., Hauch, S., and Kutyniok, G. Explaining Neural Network Decisions Is Hard. XXAI: Extending Explainable AI Beyond Deep Models and Classifiers, ICML 2020 Workshop, 2020.

Macdonald, J., Besançon, M., and Pokutta, S. Interpretable Neural Networks with Frank-Wolfe: Sparse Relevance Maps and Relevance Orderings, 11 2021. URL https://doi.org/10.5281/zenodo.5718781.

McBee, M. P., Awan, O. A., Colucci, A. T., Ghobadi, C. W., Kadom, N., Kansagra, A. P., Tridandapani, S., and Auffermann, W. F. Deep Learning in Radiology. *Academic Radiology*, 25(11):1472–1480, 2018. ISSN 1076-6332. doi: 10.1016/j.acra.2018.02.018.

Minka, T. P. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2001. AAI0803033.

Mokhtari, A., Hassani, H., and Karbasi, A. Stochastic Conditional Gradient Methods: From Convex Minimization to Submodular Maximization. *Journal of Machine Learning Research*, 21(105):1–49, 2020. URL http://jmlr.org/papers/v21/18-764.html.

Montavon, G., Samek, W., and Müller, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018. doi: 10.1016/j.dsp.2017.10.011.

Négiar, G., Dresdner, G., Tsai, A., Ghaoui, L. E., Locatello, F., Freund, R. M., and Pedregosa, F. Stochastic Frank-Wolfe for Constrained Finite-Sum Minimization. Preprint, arXiv:2002.11860, 2020.

Pokutta, S., Spiegel, C., and Zimmer, M. Deep Neural Network Training with Frank-Wolfe. Preprint, arXiv:2010.07243, 2020.

Reddi, S. J., Sra, S., Póczos, B., and Smola, A. Stochastic Frank-Wolfe methods for nonconvex optimization. In *54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1244–1251, 2016. doi: 10.1109/ALLERTON.2016.7852377.

Ribeiro, M. T., Singh, S., and Guestrin, C. ”Why Should I Trust You?”: Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

Ribeiro, M. T., Singh, S., and Guestrin, C. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K.-R. Evaluating the Visualization of What a Deep Neural Network Has Learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 11 2017. ISSN 2162-237X. doi: 10.1109/TNNLS.2016.2599820.

Shen, D., Wu, G., and Suk, H.-I. Deep Learning in Medical Image Analysis. *Annual Review of Biomedical Engineering*, 19(1):221–248, 2017. doi: 10.1146/annurev-bioeng-071516-044442.

Shen, Z., Fang, C., Zhao, P., Huang, J., and Qian, H. Complexities in Projection-Free Stochastic Non-convex Minimization. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 2868–2876. PMLR, 4 2019. URL https://proceedings.mlr.press/v89/shen19b.html.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. Preprint, arXiv:1409.1556, 2014.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. Preprint, arXiv:1312.6034, 2013.

Smilkov, D., Thorat, N., Kim, B., Viégas, F. B., and Wattenberg, M. SmoothGrad: removing noise by adding noise. Preprint, arXiv:1706.03825, 2017.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for Simplicity: The All Convolutional Net. Preprint, arXiv:1412.6806, 2015.

Szegedy, C., Toshev, A., and Erhan, D. Deep Neural Networks for Object Detection. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 26, pp. 2553–2561. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wäldchen, S., Macdonald, J., Hauch, S., and Kutyniok, G. The Computational Complexity of Understanding Binary Classifier Decisions. *J. Artif. Int. Res.*, 70:351–387, 5 2021. doi: 10.1613/jair.1.12359.

Wolfe, P. Convergence theory in nonlinear programming. In Abadie, J. (ed.), *Integer and Nonlinear Programming*. North-Holland, Amsterdam, 1970.

Yurtsever, A., Sra, S., and Cevher, V. Conditional Gradient Methods via Stochastic Path-Integrated Differential Estimator. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7282–7291. PMLR, 6 2019. URL https://proceedings.mlr.press/v97/yurtsever19b.html.

Zhang, Y., Tiňo, P., Leonardis, A., and Tang, K. A Survey on Neural Network Interpretability. Preprint, arXiv:2012.14261, 2021.

A. List of Abbreviations

Relevance Attribution Methods	
RDE	Rate-Distortion Explanations
L-RDE	Lagrangian formulation of RDE
RC-RDE	Rate-Constrained formulation of RDE
MR-RDE	Multi-Rate variant of RDE
Ord-RDE	Ordering variant of RDE
LRP	Layerwise Relevance Propagation
LIME	Local Interpretable Model-Agnostic Explanations
SHAP	Shapley Additive Explanations
Optimization Methods	
FW	Frank-Wolfe algorithm
AFW	Away-Step Frank-Wolfe algorithm
LCG	Lazified Conditional Gradients algorithm
LAFW	Lazified Away-Step Frank-Wolfe algorithm
SFW	Stochastic Frank-Wolfe algorithm
GD	Gradient Descent algorithm
PGD	Projected Gradient Descent algorithm
LMO	Linear Minimization Oracle

B. Technical Considerations

In this section, we give an overview of three deterministic variants of the Frank-Wolfe algorithm that offer potential improvements over vanilla Frank-Wolfe in terms of iteration count or runtime in certain specialized settings. Further, we specify all feasible regions considered in our experiments and present the corresponding linear minimization oracles for FW and projections for PGD.

B.1. Variations of the Frank-Wolfe Algorithm

Recall that Frank-Wolfe algorithms, at their core, solve a linear minimization oracle

$$\mathbf{v}_t = \operatorname{argmin}_{\mathbf{v}\in\mathcal{C}} \left\langle \nabla D(\mathbf{s}_t), \mathbf{v} \right\rangle , \qquad \text{(3, restated)}$$

over the feasible region $\mathcal{C}$ and then move in direction $\mathbf{v}_t$ via the update $\mathbf{s}_{t+1} = \mathbf{s}_t + \eta_t(\mathbf{v}_t - \mathbf{s}_t)$, that is a convex combination of $\mathbf{v}_t$ and the iterate $\mathbf{s}_t$.

Complementing the descriptions of the Vanilla Frank-Wolfe (FW) and Stochastic Frank-Wolfe (SFW) algorithms in Section 3 we also consider the following three deterministic Frank-Wolfe variants in our experiments.

Away-Step Frank-Wolfe (AFW) While the vanilla Frank-Wolfe algorithm can only move *towards* an extreme point of the feasible set (solution of the LMO at the current iterate), the Away-Step Frank-Wolfe algorithm (Wolfe, 1970; Guélat & Marcotte, 1986; Lacoste-Julien & Jaggi, 2015) is allowed to move *away* from some extreme points. More specifically, it maintains an *active set* of extreme points used in the previous iterations as well as a convex decomposition of the current iterate in terms of the active set. At each iteration either a standard FW step towards a new extreme point or a step away from an extreme point in the active set is taken, whichever promises a better decrease in the objective function. This can result in faster convergence (in terms of iteration count and time) but requires additional memory to store the active set.

Lazified Conditional Gradients (LCG) In some cases the evaluation of the LMO might be costly (even if it is still cheaper than a corresponding projection). In such a setting, the idea of *lazy* FW steps can help to avoid unnecessary evaluations of the LMO. Instead of exactly solving the LMO subproblem, an approximate solution that guarantees enough progress is used (Braun et al., 2019). In other words, the LMO can be replaced by a weak separation oracle (Braun et al., 2019), i.e., an oracle returning an extreme point with sufficient decrease of the linear objective or a certificate that such a point does not exist. More precisely, the algorithm maintains a cache of previous extreme points and at each iteration searches the cache for a direction that provides sufficient progress. If this is not possible, a new extreme point is obtained via the LMO and added to the cache. The lazification can result in increased performance (due to fewer LMO evaluations) but requires additional memory to store the cache of previous extreme points.

Lazified Away-Step Frank-Wolfe (LAFW) LAFW uses the same idea of a weak separation oracle as in LCG. The search for an appropriate direction providing sufficient progress is carried out over the active set of AFW.

B.2. Feasible Regions and Linear Minimization Oracles

Two different feasible sets are of importance in this work. For both, we give a brief definition and description of the corresponding linear minimization oracle below.

k-sparse polytope For $k \in [n]$, the *k-sparse polytope of radius* $\tau > 0$ is the intersection of the closed ℓ_1-ball $B_1(\tau k)$ of radius τk and the closed ℓ_∞-ball (hypercube) $B_\infty(\tau)$ of radius τ. It is the convex hull of vectors in $\mathbb{R}^n$ with exactly k non-zero entries, each taking the values τ or $-\tau$.

LMO: A valid solution $\mathbf{v}$ to (3) for the k-sparse polytope is given by the vector with exactly k non-zero entries at the components where $|\nabla D(\mathbf{s})|$ takes its k largest values. If v_j is such a non-zero entry then it is equal to $-\tau \operatorname{sign}((\nabla D(\mathbf{s}))_j)$. The complexity of finding this solution is $\mathcal{O}(n \log k)$.

Projection: Gupta et al. (2010) propose an $\mathcal{O}(n)$ algorithm for projections onto ℓ_1-balls with box-constraints, extending the work of Duchi et al. (2008) on efficient projections onto ℓ_1-balls. It is based on linear time median finding, see e.g. (Cormen et al., 2009). A slightly simplified $\mathcal{O}(n \log n)$ variant based on sorting is used in our implementation.

More relevant to us is the following variation:

Non-negative k-sparse polytope: For $k \in [n]$ and $\tau > 0$ the *non-negative k-sparse polytope of radius τ* is defined as the intersection of the k-sparse polytope of radius τ with the non-negative orthant $\mathbb{R}^n_{\geq 0}$.

LMO: A valid solution $\mathbf{v}$ to (3) for the non-negative k-sparse polytope is given by the vector with at most k non-zero entries at the components where $\nabla D(\mathbf{s})$ is negative and takes its k smallest values (thus largest in magnitude as above). If $\nabla D(\mathbf{s})$ has fewer than k negative entries, then $\mathbf{v}$ has fewer than k non-zero entries. If v_j is a non-zero entry then it is equal to τ. As above the complexity of finding this solution is $\mathcal{O}(n \log k)$.

Projection: The same algorithm for projections onto ℓ_1-balls with box-constraints as above can be used.

The feasible region $\mathcal{C} = \{\mathbf{s} \in [0,1]^n \ : \ \|\mathbf{s}\|_1 \leq k\}$ for the (RC-RDE) problem is the non-negative k-sparse polytope of radius $\tau = 1$.

Birkhoff polytope The *Birkhoff polytope B_n* is the set of doubly-stochastic $(n \times n)$-matrices. It is the convex hull of the set of $(n \times n)$-permutation matrices.

LMO: The Birkhoff polytope arises in matching and ranking problems. Linear minimization over B_n results in a linear program, which can be solved with $\mathcal{O}(n^3)$ complexity using the Hungarian method (Kuhn, 1955) implemented in the `Hungarian.jl` package. Linear minimization can also be performed using the standard or network simplex algorithms, opening the possibility for optimized and potentially parallelizable implementations. In our experiments we found that the LMO was nonetheless more efficient, both in terms of runtime and memory footprint, when using the Hungarian algorithm compared to off-the-shelf simplex solvers.

Projection: To the best of our knowledge, there is no exact projection method specific to the Birkhoff polytope. An approximate method based on the Douglas-Ratchford splitting algorithm was proposed by Combettes & Pokutta (2021). Its complexity to achieve ϵ-convergence is $\mathcal{O}(n^2 c^2/\epsilon^2)$ where c is not known a-priory and depends on the distance of the initial guess for the algorithm and a fixed point of the proximal operator evaluated in each iteration.

The set B_n is used as the feasible region for the (Ord-RDE) problem.

B.3. Computational Setup

We give a brief specification of our computational setup. All experiments were run on computer cluster nodes with a Nvidia Quadro RTX 6000 GPU and an AMD EPYC 7262 or AMD EPYC 7302P CPU. We use `Julia (v1.6.1)` and the `FrankWolfe.jl (v0.1.8)` package for all variants of FW algorithms. The `Python (v3.7.8)` packages `Tensorflow (v1.15.0)` and `Keras (v2.2.4)` are used for the neural network classifiers and computation of gradients through automatic differentiation. The interactions between `Julia` and `Python` are handled through the `PyCall.jl (v1.92.3)` package.

C. Additional Computational Results

Figure 9 shows a comparison of results obtained using the different FW variants for solving (RC-RDE) and (MR-RDE) for the MNIST experiment example image from Figure 3. All FW variants yield similar results and are robust across varying rates, in the sense that solutions for larger rates add additional features to the relevant set without significantly modifying the features that were already considered relevant at smaller rates. Figures 11 and 12 complement Figures 4 and 6 and show shows the relevance ordering test results for all FW variants for the MNIST and STL-10 experiment respectively.

A comparison of runtimes for the MNIST experiment and the corresponding numbers of iterations that are taken until the termination criterion $\langle \mathbf{s}_t - \mathbf{v}_t, \nabla D(\mathbf{s}_t) \rangle < \varepsilon = 10^{-7}$ is reached for the different FW variants is shown in Figure 10(a). We observe that AFW converges fastest for small rates k, while all variants perform similarly at higher rates (with slight advantages of FW and LCG over AFW and LAFW). This is due to a reduced number of iterations of AFW and LAFW at small rates. On the other hand, the increased runtime of the active-set methods AFW and LAFW can be explained by the fact that each vertex in the active set is a sparse vector with more and more non-zero components as the rate increases. Hence, active set operations require more arithmetic operations at higher rates. All methods reach the maximum

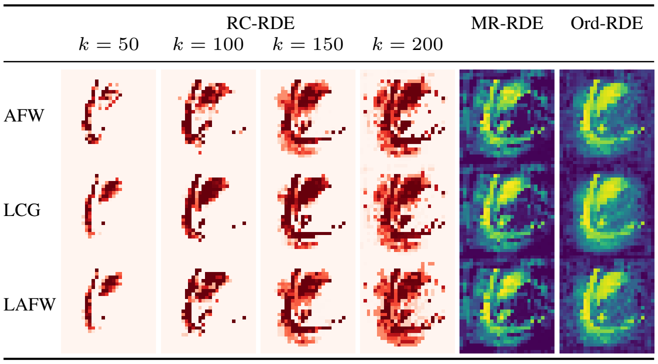

Figure 9. Additional relevance mappings for the MNIST image from Figure 3 classified as *digit six* by the network. Multi-rate solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but as component orderings from least relevant (blue) to most relevant (yellow).

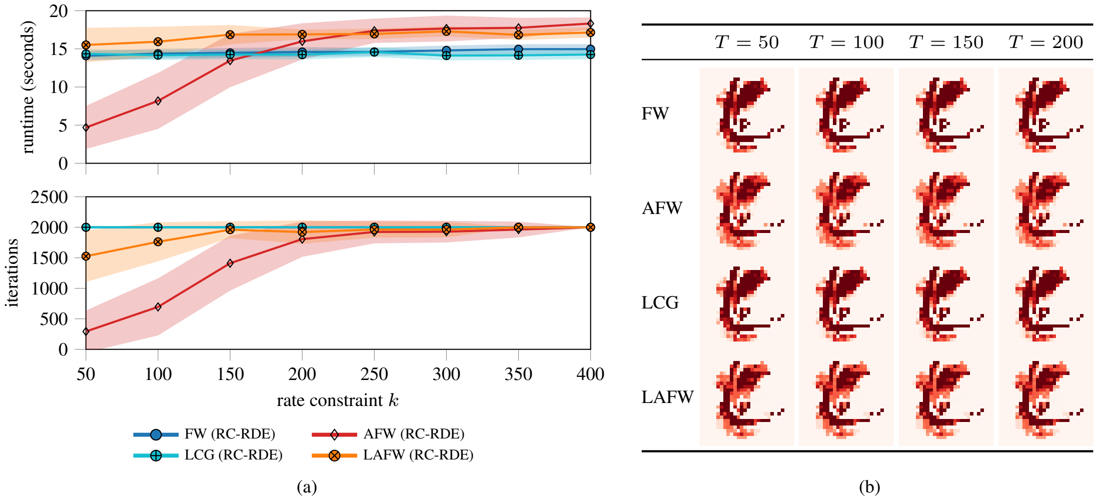

Figure 10. (a) Average runtimes (top) and number of iterations until convergence (bottom) of the considered FW variants for (RC-RDE) on MNIST at different rates. An average result over 50 images from the test set is shown (shaded regions mark ± standard deviation). (b) Relevance mappings obtained at rate $k = 150$ after different maximal numbers of iterations. Results for the same MNIST image from Figure 3 classified as *digit six* by the network are shown. All methods were converged effectively already after only $T = 50$ iterations.

number of $T = 2000$ iterations before satisfying the termination criterion at high rates. This can be explained by the fact that the termination threshold $\varepsilon = 10^{-7}$ is chosen quite conservatively to ensure convergence of all methods on all instances. However, we observe that all methods typically converge much faster to a satisfactory solution. Figure 10(b) shows (RC-RDE) solutions at a single exemplary rate $k = 150$ for the MNIST image from Figure 3 after $T = 50$, 100, 150, and 200 iterations. The solutions are visually indistinguishable for all methods, confirming that they were already mostly converged after only 50 iterations. We do not show a comparable analysis of runtimes and iteration counts for the STL-10 experiment. Due to higher computational costs, the STL-10 mappings were computed in parallel on different machines. Hence, no runtimes that are directly comparable between the different methods are available.

The relevance-ordering problem (Ord-RDE) can be solved with a stochastic Frank-Wolfe algorithm (as shown in Section 4) or with deterministic Frank-Wolfe algorithms if the number n of terms in the objective function is not too large. For the MNIST dataset, both approaches are feasible. We compare the relevance-ordering comparison test results of all FW variants in Figure 13. The SFW result is the same as in Figure 4. We observe that all variants perform well and similarly for very low and high rates. For rates between 20% and 80% of the total number of components SFW has an advantage over FW and LCG which in turn perform slightly better than AFW and LAFW.

Further, we compare different hyperparameter configurations for SFW regarding the batch sizes b_t and momentum factors

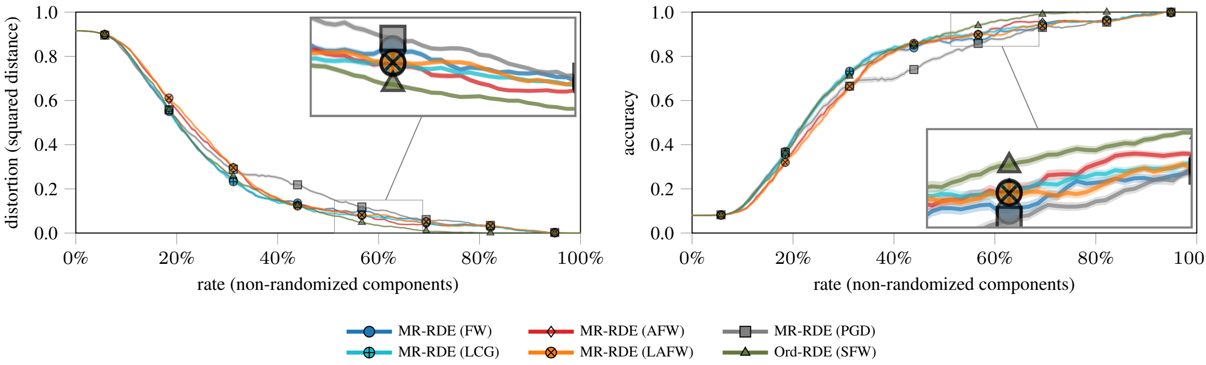

Figure 11. Relevance ordering test results for MNIST for RDE with different FW variants and PGD. An average result over 50 images from the test set (5 images per class) and 512 noise input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure 4.

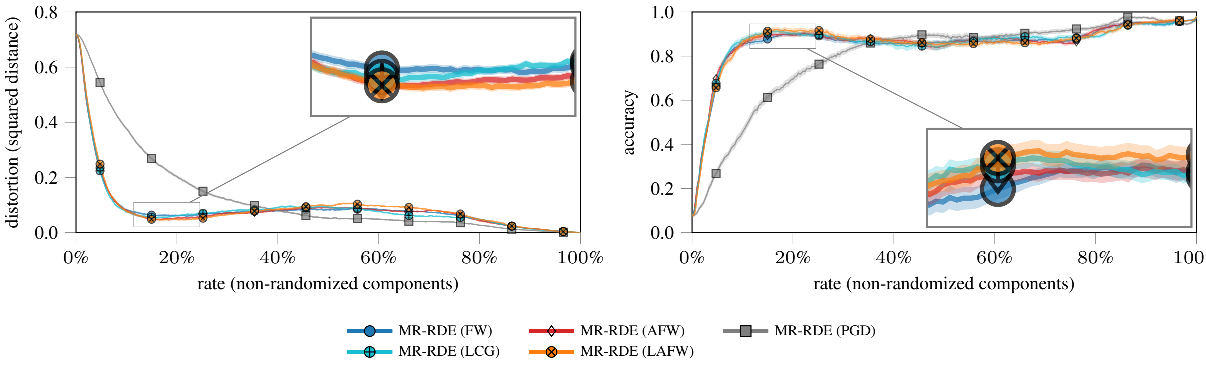

Figure 12. Relevance ordering test results for STL-10 for RDE with different FW variants and PGD. An average result over 50 images from the test set (5 images per class) and 64 noise input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure 6.

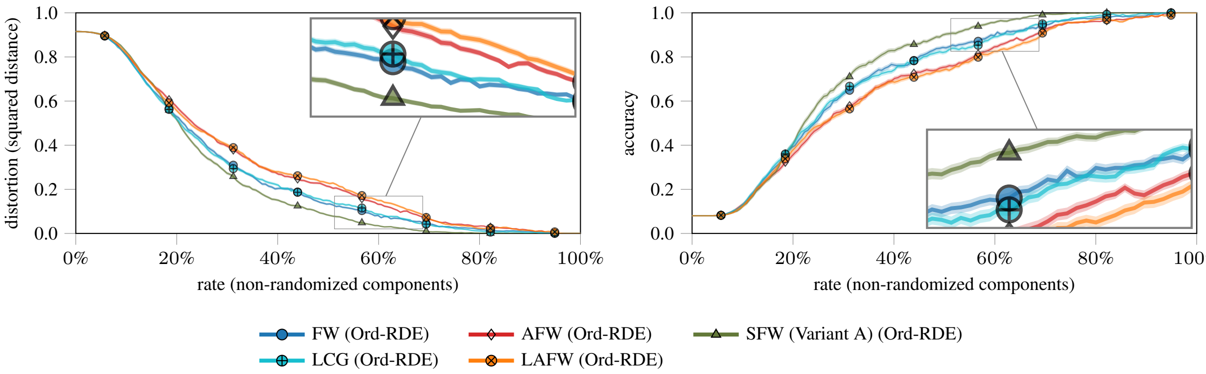

Figure 13. Relevance ordering test results for (Ord-RDE) on the MNIST dataset for all considered FW variants. An average result over 50 images from the test set (5 images per class) and 512 noise input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure 4, which contains the same SFW result.

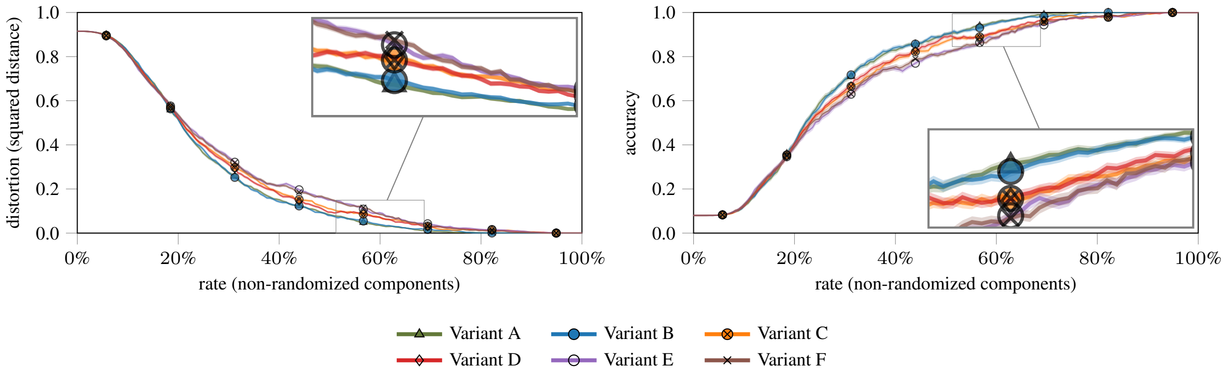

Figure 14. Relevance ordering test results for (Ord-RDE) on MNIST for different variants of SFW. An average result over 50 images from the test set (5 images per class) and 512 noise input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure 4, which contains the same SFW results with constant batch size and no momentum.

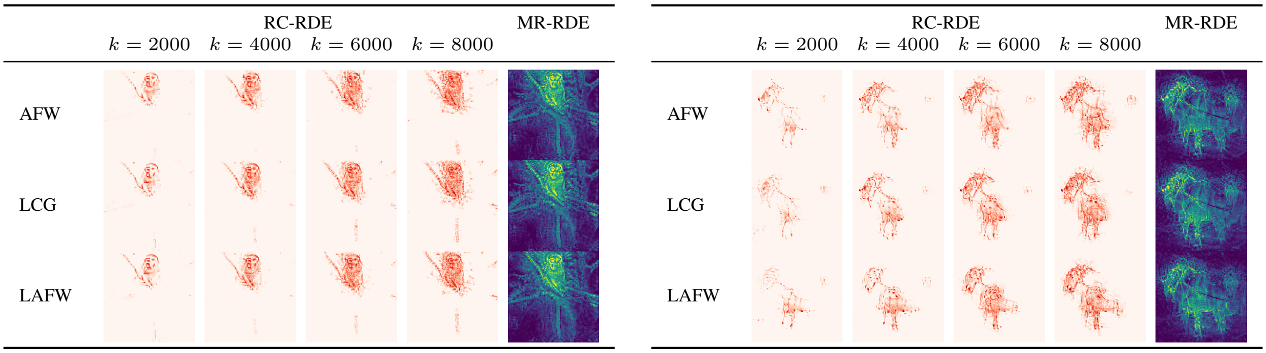

Figure 15. Relevance mappings for STL-10 images classified as *monkey* and *horse* by the network. This complements Figures 7 and 8.

ρ_t. Hazan & Luo (2016) suggest a linearly increasing[11] batch size and Mokhtari et al. (2020) propose a momentum factor $\rho_t = 1 - 4/(8+t)^{\frac{2}{3}}$ approaching one. We consider the following combinations:

	Variant A	Variant B	Variant C	Variant D	Variant E	Variant F
momentum	$\rho_t = 0$	$\rho_t = 0$	$\rho_t = \frac{1}{2}$	$\rho_t = \frac{1}{2}$	$\rho_t = 1 - \frac{4}{(8+t)^{\frac{2}{3}}}$	$\rho_t = 1 - \frac{4}{(8+t)^{\frac{2}{3}}}$
batch size	$b_t = 40$	$b_t = \min\{40+t, 100\}$	$b_t = 40$	$b_t = \min\{40+t, 100\}$	$b_t = 40$	$b_t = \min\{40+t, 100\}$

The relevance-ordering comparison test results are shown in Figure 14. For reference Variant A corresponds to the SFW result also shown in Figure 4 and 13. We observe, that all variants perform well and similarly across all rates. The batch size has negligible effect in this experiment, while momentum yields no advantage. In particular, both configurations without any momentum perform best. Hence, we show only Variant A (no momentum, constant batch size) in all other experiments.

D. Additional STL-10 Relevance Maps

We complement Figures 7 and 8 by Figures 15–23 showing additional examples of relevance maps for images of different classes from the STL-10 dataset as well as additional results for all FW variants for the images from Figures 7 and 8. The colormap indicates positive relevance as red and negative relevance as blue. All multi-rate solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but as component orderings from least relevant (blue) to most relevant (yellow). All (L-RDE) solutions and results for the other comparison methods are provided by Macdonald et al. (2019).

[11] We limit the batch size growth up to a maximal size $b_{\text{max}} = 100$ in our experiments.

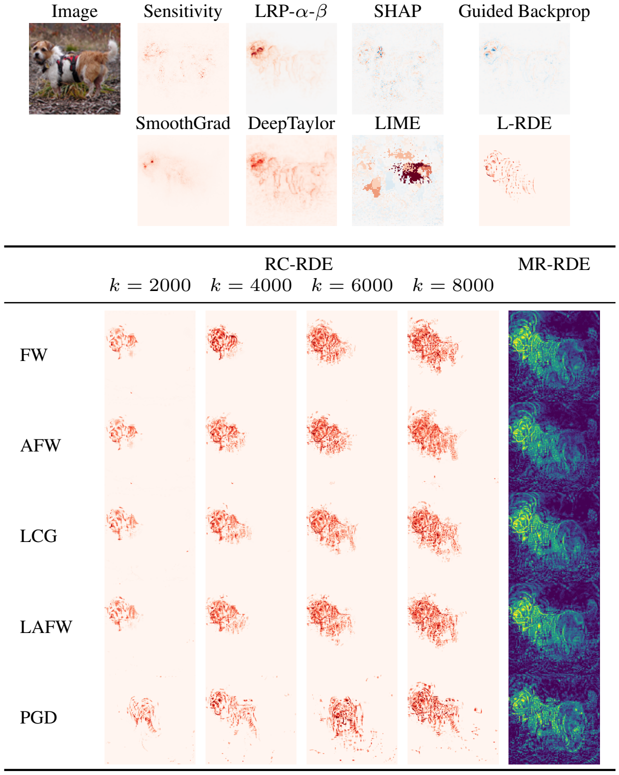

Figure 16. Relevance mappings for an STL-10 image classified as *dog* by the network.

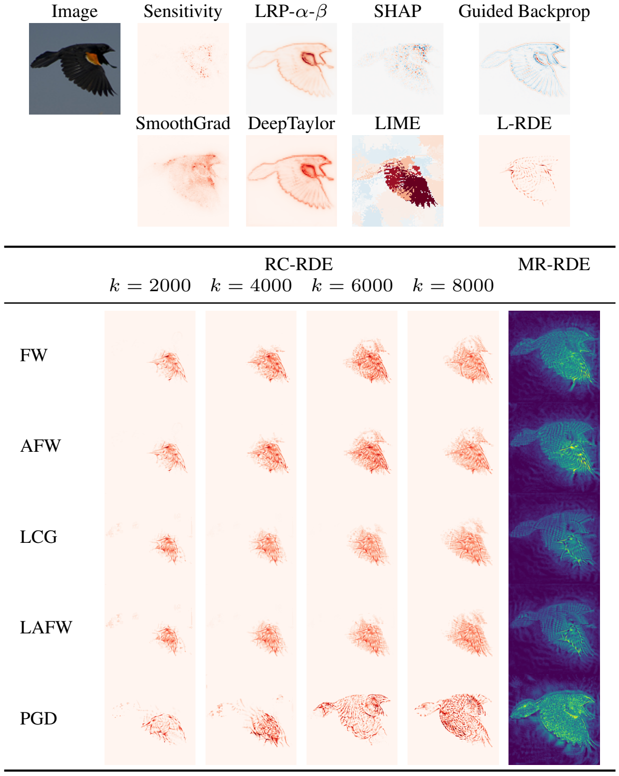

Figure 17. Relevance mappings for an STL-10 image classified as *bird* by the network.

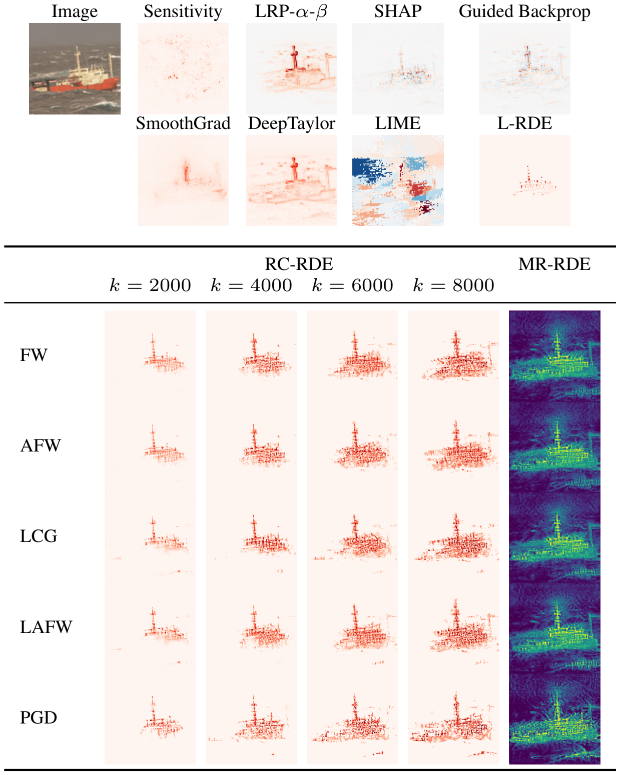

Figure 18. Relevance mappings for an STL-10 image classified as *ship* by the network.

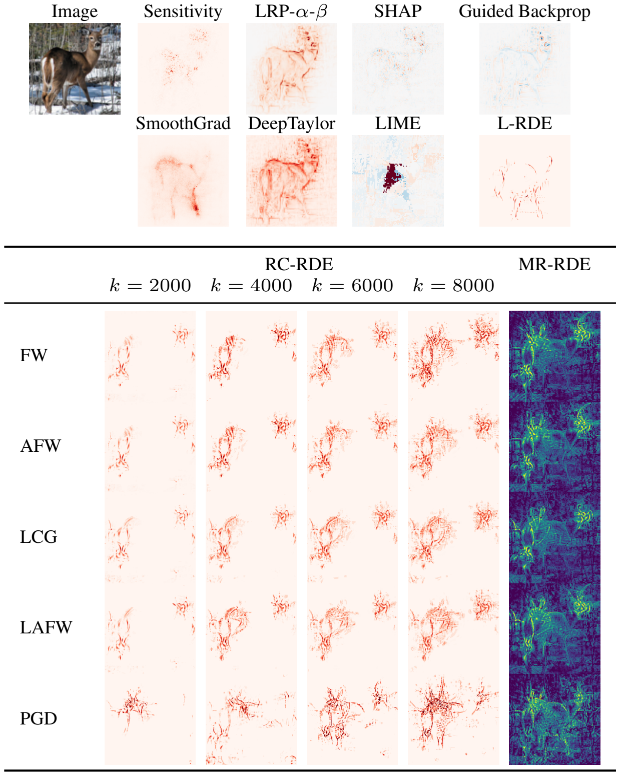

Figure 19. Relevance mappings for an STL-10 image classified as *deer* by the network.

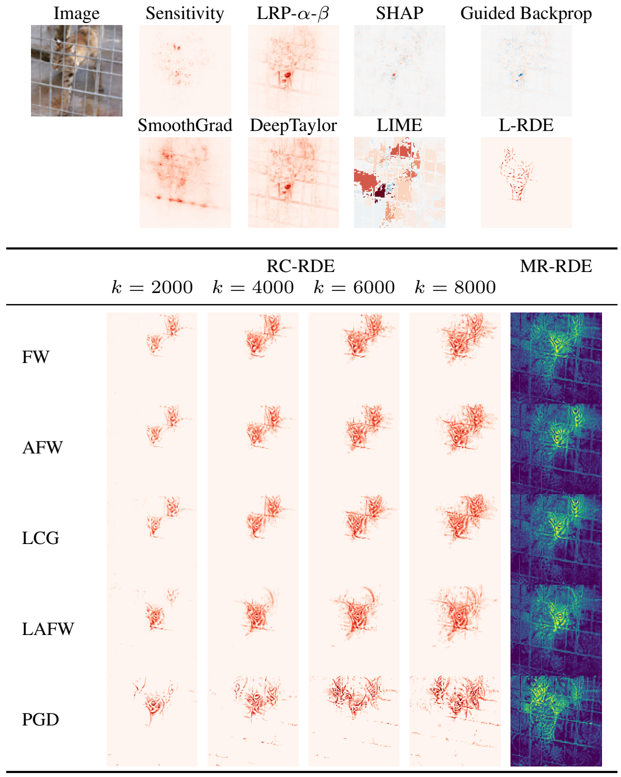

Figure 20. Relevance mappings for an STL-10 image classified as *cat* by the network.

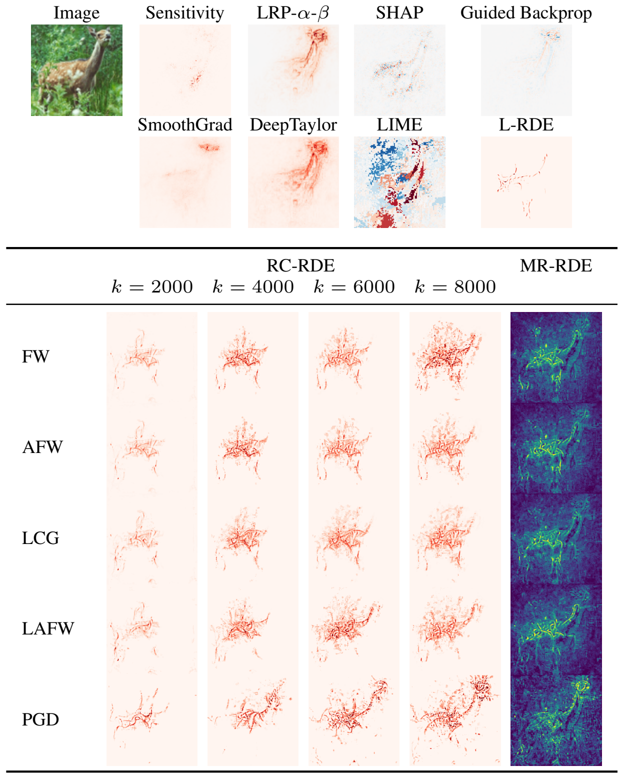

Figure 21. Relevance mappings for an STL-10 image classified as *deer* by the network.

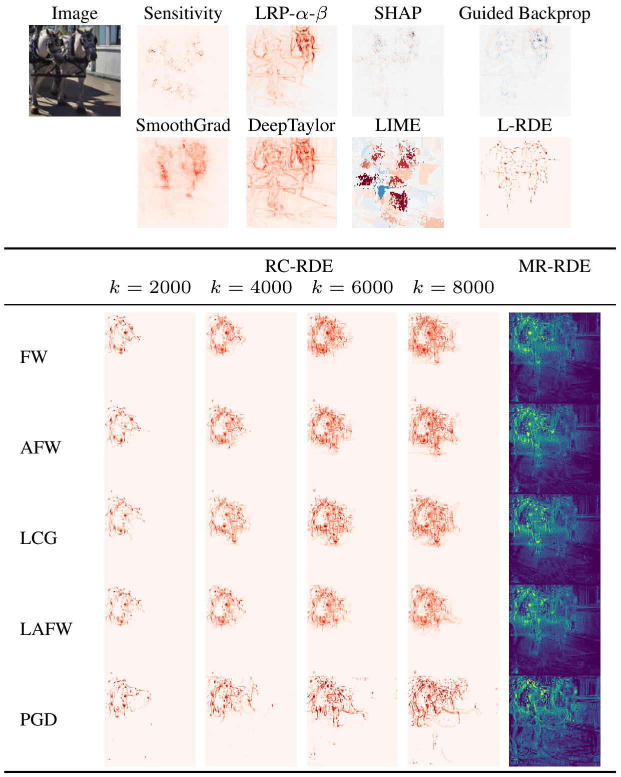

Figure 22. Relevance mappings for an STL-10 image classified as *horse* by the network.

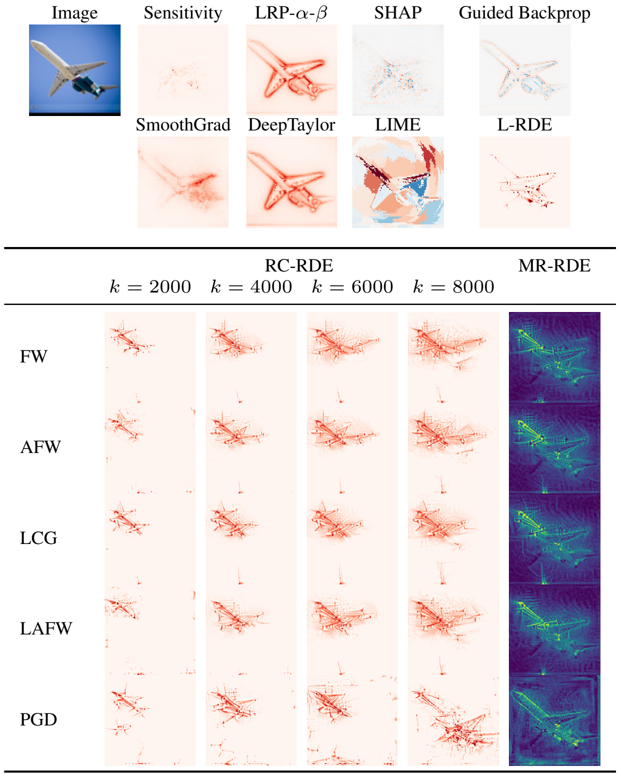

Figure 23. Relevance mappings for an STL-10 image classified as *airplane* by the network.