A Rate-Distortion Framework for Explaining Neural Network Decisions

Jan Macdonald, Stephan Wäldchen, Sascha Hauch
Technische Universität Berlin
{macdonald, stephanw, hauch}@math.tu-berlin.de

Gitta Kutyniok
Technische Universität Berlin
University of Tromsø
kutyniok@math.tu-berlin.de

Abstract

We formalise the widespread idea of interpreting neural network decisions as an explicit optimisation problem in a rate-distortion framework. A set of input features is deemed relevant for a classification decision if the expected classifier score remains nearly constant when randomising the remaining features. We discuss the computational complexity of finding small sets of relevant features and show that the problem is complete for $\mathsf{NP}^{\mathsf{PP}}$, an important class of computational problems frequently arising in AI tasks. Furthermore, we show that it even remains NP-hard to only approximate the optimal solution to within any non-trivial approximation factor. Finally, we consider a continuous problem relaxation and develop a heuristic solution strategy based on assumed density filtering for deep ReLU neural networks. We present numerical experiments for two image classification data sets where we outperform established methods in particular for sparse explanations of neural network decisions.

1 Introduction

Traditional machine learning models such as linear regression, decision trees, or k-nearest neighbours allow for a straight-forward human interpretation of the model prediction. In contrast, the reasoning of highly nonlinear and parameter-rich neural networks remains generally inaccessible. Recent years have seen progress on this front with the introduction of multiple explanation models for deep neural networks [2, 12, 18, 21, 22, 27]. These models provide additional information to a prediction by assigning importance values to individual input features . The evaluation of the soundness of these methods has so far mostly been limited to comparison with human intuition as to which features are important. Notable exceptions are Shapley values [20] that draw justification from game theoretic aspects, as well as the proposition of pixel-flipping to numerically compare explanation methods [19]. However, there is yet no formal definition of what it means for an input feature to be relevant for a classification decision.

In this paper we introduce a rigorous approach to obtain interpretable neural network decisions. More precisely, in Section 2 we formulate the problem of determining the most relevant components of an input signal for a classifier prediction as an optimisation problem in a rate-distortion framework. We show in a worst-case analysis in Section 3 that this problem is generally hard to solve and to approximate, which justifies the use of heuristic methods. In Section 4 we propose a problem relaxation together with a heuristic solution strategy for deep feed forward neural networks. Finally, we present numerical experiments and compare different explanation methods for two image classification data sets in Section 5.

Notation Throughout, $d \in \mathbb{N}$ is the dimension of the signal domain, $\mathbf{x} \in [0,1]^d$ is an arbitrary fixed input signal, and $\Phi\colon [0,1]^d \to [0,1]$ is a classifier function for a signal class $\mathcal{C} \subseteq [0,1]^d$. The function Φ can for example be described by a neural network. The classification score $\Phi(\mathbf{x})$ represents the classifiers prediction on how likely it is that $\mathbf{x}$ belongs to the class $\mathcal{C}$. We set $[d] = \{1, \dots, d\}$ and for a subset $S \subseteq [d]$ denote by $\mathbf{x}_S$ the restriction of $\mathbf{x}$ to components indexed by S. Finally, $\mathbf{1}_d \in \mathbb{R}^d$ is a vector of all ones, $\operatorname{diag}(\mathbf{x})$ the diagonal matrix with entries given by $\mathbf{x}$, and $\odot$ (resp. $\oslash$) the component-wise Hadamard product (resp. quotient) of vectors or matrices of the same dimensions. We write $\mathbf{x}^2 = \mathbf{x} \odot \mathbf{x}$ and consider univariate functions applied to vectors to act component-wise.

2 Rate-Distortion viewpoint

We formulate the task of explaining the classifier prediction $\Phi(\mathbf{x})$ as finding a partition of the components of $\mathbf{x} \in [0,1]^d$ into a subset $S \subseteq [d]$ of *relevant* components and its complement S^c of *non-relevant* components. The partition should be chosen in a way such that fixing the relevant components already determines the classifier output for almost all possible assignments to the non-relevant ones. More precisely, let $\mathcal{V}$ be a probability distribution on $[0,1]^d$ and $\mathbf{n} \sim \mathcal{V}$ a random vector. We define the *obfuscation* of $\mathbf{x}$ with respect to S and $\mathcal{V}$ as a random vector $\mathbf{y}$ that is deterministically defined on S as $\mathbf{y}_S = \mathbf{x}_S$ and distributed on the complement according to $\mathbf{y}_{S^c} = \mathbf{n}_{S^c}$. We write $\mathcal{V}_S$ for the resulting distribution of $\mathbf{y}$. Having only knowledge about the signal on S and filling in the rest of the signal randomly will mostly preserve the class prediction if $\mathbf{x}_S$ contains the information that was relevant for the classifier decision. We measure the change in the classifier prediction using the squared distance. The expected distortion of S with respect to Φ, $\mathbf{x}$, and $\mathcal{V}$ is defined as

$$D(S) = D(S, \Phi, \mathbf{x}, \mathcal{V}) = \mathbb{E}_{\mathbf{y} \sim \mathcal{V}_S}\left[\frac{1}{2}\left(\Phi(\mathbf{x}) - \Phi(\mathbf{y})\right)^2\right].$$

We naturally arrive at a rate-distortion trade-off that intuitively gives us a measure of relevance. The terminology is borrowed from information theory where rate-distortion is used to analyse lossy data compression. In that sense, the set of relevant components is a compressed description of the signal and the expected deviation from the classification score is a measure for the reconstruction error. We define the rate-distortion function as

$$R(\epsilon) = \min\left\{\, |S| \,:\, S \subseteq [d],\, D(S) \leq \epsilon \,\right\}. \tag{1}$$

The smallest set S that ensures a limited distortion will be composed of the most relevant input components. This rate-distortion function will also allow us later to compare the performance of different explanation models.

We now want to discuss the difficulty of finding such a set. Note, that the trivial choice of setting $S = [d]$ ensures zero distortion. We show that for distortion limits greater than zero one cannot systematically find a set of relevant components that is significantly smaller than the trivial set. This specifically holds when Φ is represented by a neural networks which is the case we are particularly interested in. We derive our hardness results for the special case of Boolean circuits representable by ReLU neural networks of moderate size.[1]

3 Complexity theoretic analysis

Let us for now consider the special case of binary input signals $\mathbf{x} \in \{0,1\}^d$ and classifier functions $\Phi : \{0,1\}^d \to \{0,1\}$ described as Boolean circuits, as well as the uniform distribution over binary vectors, i.e. $\mathcal{V} = \mathcal{U}\left(\{0,1\}^d\right)$.

[1]The depth can be chosen constant and the total number of neurons polynomial in d.

3.1 Discrete problem formulation

We call a subset $S \subseteq [d]$ a *δ-relevant* set for Φ and $\mathbf{x}$ if $\mathbb{P}_{\mathbf{y}}\left(\Phi(\mathbf{y}) = \Phi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S\right) \geq \delta$. We can assert that a set S is δ-relevant if and only if it has limited distortion $D(S) \leq \frac{1-\delta}{2}$. Therefore calculating the rate-distortion function is essentially the same task as finding minimal relevant sets.

Definition 3.1. For $\delta \in [0,1]$ we define the RELEVANT-INPUT problem as follows.
GIVEN: $\Phi\colon \{0,1\}^d \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^d$, and $k \in [d]$.
DECIDE: Does there exist an $S \subseteq [d]$ with $|S| \leq k$ such that S is δ-relevant for Φ and $\mathbf{x}$?

The optimisation version associated to this decision problem can be defined in the standard way.

Definition 3.2. For $\delta \in [0,1]$ we define the MIN-RELEVANT-INPUT problem as follows.
GIVEN: $\Phi\colon \{0,1\}^d \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^d$.
MINIMISE: $k \in [d]$ such that there exists an $S \subseteq [d]$ with $|S| \leq k$ that is δ-relevant for Φ and $\mathbf{x}$.

Then the following hardness result holds [26].

Theorem 3.3. *For $1/2 \leq \delta < 1$ the* RELEVANT-INPUT *problem is* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

The class $\mathsf{NP}^{\mathsf{PP}}$ is the class of all problems decidable by a non-deterministic Turing machine with access to an oracle for problems in PP [9]. The class $\mathsf{NP}^{\mathsf{PP}}$ appears frequently in artificial intelligence tasks such as optimisation under uncertainty [11] and is assumed significantly harder than either NP or PP [6, 16]. This means also that solving MIN-RELEVANT-INPUT is an extremely difficult problem. In practice it would often suffice to solve it only approximately and one might hope that efficient approximation algorithms exist. Hence, a more practically relevant result is the following.

Theorem 3.4. *Assume $\mathsf{P} \neq \mathsf{NP}$ then for any $\alpha \in (0,1)$ there is no polynomial time approximation algorithm for* MIN-RELEVANT-INPUT *with an approximation factor of $d^{1-\alpha}$.*

We give a full proof in the following section and want to remark that this is a simplification of an even a slightly stronger result shown in [26].

3.2 Inapproximability

An approximation algorithm for MIN-RELEVANT-INPUT has an approximation factor $c \geq 1$ if it finds an approximate solution k such that $k^* \leq k \leq ck^*$ for all problem instances, where k^* denotes the respective exact solution. Choosing the trivial solution $S = [d]$, thus considering all components as relevant, results in a factor d. Theorem 3.4 states that it is generally hard to achieve better factors.

The proof proceeds in two steps. First, we introduce a gapped version of the decision problem and show that it is NP-hard. Second, we show that the gapped problem would be in P if there exists a good polynomial time approximation algorithm for MIN-RELEVANT-INPUT.

Definition 3.5. For $\delta \in [0,1]$ we define the AUXILLIARY PROBLEM (AP) as follows.
GIVEN: $\Phi\colon \{0,1\}^d \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^d$, and $k, m \in \mathbb{N}$, $1 \leq k \leq m \leq d$.
DECIDE: Which of the two options (if any) holds:

Yes-instances: There exists $S \subseteq [d]$ with $|S| \leq k$ and S is δ-relevant for Φ and $\mathbf{x}$.

No-instances: All $S \subseteq [d]$ with $|S| \leq m$ are not δ-relevant for Φ and $\mathbf{x}$.

The restriction to the case $k = m$ is exactly the RELEVANT-INPUT problem. However here we also allow the case $k < m$ with a gap in the set sizes. It might happen that none of the two options hold. In this case we consider any answer to be acceptable.

Lemma 3.6. *For $\delta \in (0,1)$ we have* SAT $\preceq_p$ AP*, in particular in this case* AP *is* NP*-hard.*

Proof. Let $\Phi\colon \{0,1\}^d \to \{0,1\}$ be a SAT instance. We will construct $\{\Phi', \mathbf{x}', k', m'\}$ that is a *Yes*-instance for AP if and only if Φ is a *Yes*-instance for SAT. Let $q = \lceil \log_2(d) - \log_2(1-\delta) \rceil$ and $p = \lfloor -\log_2(\delta) \rfloor + 1$. We set $k' = dq$, $m' \geq k'$ arbitrary but at most polynomial in d. Also let $\Phi'\colon \{0,1\}^{d \times q} \times \{0,1\}^{m'+p} \to \{0,1\}$ be given by

$$\Phi'(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(q)}, \mathbf{v}) = \Phi\left(\bigwedge_{j=1}^{q} \mathbf{u}^{(j)}\right) \vee \left(\bigwedge_{i=1}^{m'+p} v_i\right),$$

where each $\mathbf{u}^{(j)} \in \{0,1\}^d$ and the conjunction within Φ is component-wise, and set $\mathbf{x}' = \mathbf{1}_{dq+m'+p}$. This is a polynomial time construction. By the choice of Φ' and $\mathbf{x}'$ we guarantee $\Phi'(\mathbf{x}') = 1$ regardless of the satisfiability of Φ. In the following we denote $\mathbf{U} = (\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)})$.

Necessity: Let Φ be a *Yes*-instance for SAT. Thus, there exists $\mathbf{x} \in \{0,1\}^d$ with $\Phi(\mathbf{x}) = 1$. Let $S = \{\, i \in [d] : x_i = 1 \,\}$ and $S' = S \times [q]$. Denote $A(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)}) = \bigwedge_{(i,j) \in S'} u_i^{(j)}$. Clearly, $|S'| \leq k'$. Further, S' is δ-relevant for Φ' and $\mathbf{x}'$ if $\mathbb{P}_{(\mathbf{U},\mathbf{v})}\left(\Phi'(\mathbf{U},\mathbf{v}) \,|\, A(\mathbf{U})\right) \geq \delta$. We have

$$\mathbb{P}_{(\mathbf{U},\mathbf{v})}\left(\Phi'(\mathbf{U},\mathbf{v}) \,|\, A(\mathbf{U})\right) \geq \mathbb{P}_{\mathbf{U}}\left(\bigwedge_{j=1}^{q} \mathbf{u}^{(j)} = \mathbf{x} \,\middle|\, A(\mathbf{U})\right).$$

From this, with a union bound, we obtain

$$\mathbb{P}_{\mathbf{U}}\left(\bigwedge_{j=1}^{q} \mathbf{u}^{(j)} = \mathbf{x} \,\middle|\, A(\mathbf{U})\right) = 1 - \mathbb{P}_{\mathbf{U}}\left(\exists i \in S^c : \bigwedge_{j=1}^{q} u_i^{(j)}\right) \geq 1 - |S^c| 2^{-q} \geq \delta,$$

which shows that $\{\Phi', \mathbf{x}', k', m'\}$ is a *Yes*-instance for AP.

Sufficiency: Now conversely let Φ be a *No*-instance for SAT. Then for any $S' \subseteq [dq + m' + p]$ with $|S'| \leq m'$ we have

$$\begin{aligned}\mathbb{P}_{\mathbf{y}}\left(\Phi'(\mathbf{y}) = \Phi'(\mathbf{x}') \,|\, \mathbf{y}_{S'} = \mathbf{x}'_{S'}\right) &= \mathbb{P}_{(\mathbf{U},\mathbf{v})}\left(\bigwedge_{i=1}^{m'+p} v_i \,\middle|\, (\mathbf{U},\mathbf{v})_{S'} = \mathbf{1}\right) \\ &\leq 2^{-(m'+p-|S'|)} < \delta,\end{aligned}$$

showing that S' is not δ-relevant for Φ' and $\mathbf{x}'$. Hence, $\{\Phi', \mathbf{x}', k', m'\}$ is a *No*-instance for AP. $\square$

Finally, we can prove the inapproximability of the MIN-RELEVANT-INPUT problem.

Proof of Theorem 3.4. We show that the existence of a polynomial time approximation algorithm for MIN-RELEVANT-INPUT with approximation factor $d^{1-\alpha}$ would allow us to decide AP in polynomial time for certain instances. These can be chosen as in the proof of Lemma 3.6, which in turn implies that we could decide SAT in polynomial time. This is only possible if $\mathsf{P} = \mathsf{NP}$.

Let $\Phi\colon \{0,1\}^d \to \{0,1\}$ be a SAT instance and $\{\Phi', \mathbf{x}', k', m'\}$ an equivalent AP instance as in the proof of Lemma 3.6. We have seen that there is some freedom in the choice of m' as long as it satisfies $k' \leq m'$ and is at most polynomial in d. We choose $m' = \left\lceil \max(2k'({k'}^{1-\alpha} + p^{1-\alpha}), (2k')^{\frac{1}{\alpha}} + 1) \right\rceil$ with $p = \lfloor -\log_2(\delta) \rfloor + 1$ as before. Recall that $k' = dq$ with $q = \lceil \log_2(d) - \log_2(1-\delta) \rceil$, so clearly m' is polynomial in d and $k' \leq m'$. Further, we have $m' > (2k')^{\frac{1}{\alpha}}$ so $1 - k'{m'}^{-\alpha} > \frac{1}{2}$ and therefore

$$m'(1 - k'{m'}^{-\alpha}) > \frac{m'}{2} \geq k'({k'}^{1-\alpha} + p^{1-\alpha}).$$

Now let $d' = k' + m' + p$ denote the number of variables of Φ'. By the subadditivity of the map $z \mapsto z^{1-\alpha}$, we finally obtain

$$k'{d'}^{1-\alpha} = k'(k' + m' + p)^{1-\alpha} \leq k'\left({k'}^{1-\alpha} + {m'}^{1-\alpha} + p^{1-\alpha}\right) < m'.$$

It remains to show that an AP instance with $m' > k'{d'}^{1-\alpha}$ can be decided by an approximation algorithm for MIN-RELEVANT-INPUT with approximation factor ${d'}^{1-\alpha}$. Assume such an algorithm exists and let k be a solution produced by the algorithm. Then for the true optimal solution k^* we have $k^* \leq k \leq {d'}^{1-\alpha} k^*$.

Firstly, assume that $\{\Phi', \mathbf{x}', k', m'\}$ is a *Yes*-instance for AP. Then there exists a δ-relevant set of size k'. However, notice that no set smaller than k^* can be δ-relevant. This implies $k^* \leq k'$ and therefore $k \leq {d'}^{1-\alpha} k' < m'$.

Secondly, assume that $\{\Phi', \mathbf{x}', k', m'\}$ is a *No*-instance for AP. Then all sets of size at most m' are not δ-relevant. But there exists a δ-relevant set of size k^*. This implies $k \geq k^* > m'$.

Altogether, checking whether $k < m'$ or $k > m'$ decides $\{\Phi', \mathbf{x}', k', m'\}$. $\square$

Theorem 3.4 shows that no efficient approximation algorithm for MIN-RELEVANT-INPUT exists (unless $\mathsf{P} = \mathsf{NP}$). Either we have to resort to heuristics, or introduce stronger restrictions on the problem. We choose the former and present a general heuristic for neural networks. To this end, we also further relax the problem formulation to a continuous setting.

4 Problem relaxation and solution heuristic

The problem class $\mathsf{NP}^{\mathsf{PP}}$ already gives a hint of what difficulties we have to overcome. First, we need an efficient way to judge whether a chosen set leads to small expected distortion. This amounts to calculating expectation values. Secondly, we need to optimise over all feasible sets — a combinatorial optimisation problem. We propose a heuristic solution for both hurdles when the classifier Φ is a deep neural network.

4.1 Neural network functions

Let $L \in \mathbb{N}$ denote the number of layers of the neural network, $d_1, \dots, d_{L-1} \in \mathbb{N}$ and $d_0 = d$, $d_L = 1$. Further let $(\mathbf{W}_1, \mathbf{b}_1), \dots, (\mathbf{W}_L, \mathbf{b}_L)$ with $\mathbf{W}_i \in \mathbb{R}^{d_i \times d_{i-1}}$, $\mathbf{b}_i \in \mathbb{R}^{d_i}$ for $i \in [L]$ be the weight matrices and bias vectors of an L-layer feed forward neural network. We then consider functions of the form

$$\Phi(\mathbf{x}) = \mathbf{W}_L \varrho(\mathbf{W}_{L-1} \varrho(\dots \varrho(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \dots) + \mathbf{b}_{L-1}) + \mathbf{b}_L.$$

In the following we consider the rectified linear unit (ReLU) activation function $\varrho(x) = \max\{0, x\}$.

4.2 Continuous problem relaxation

To address the combinatorial optimisation problem we make use of the following relaxation. Instead of binary relevance decisions (*relevant* versus *non-relevant*) encoded by the set S, we allow for a continuous relevance score for each component, encoded by a vector $\mathbf{s} \in [0, 1]^d$. We redefine the obfuscation of $\mathbf{x}$ with respect to $\mathbf{s}$ as a component-wise convex combination

$$\mathbf{y} = \mathbf{x} \odot \mathbf{s} + \mathbf{n} \odot (\mathbf{1} - \mathbf{s}) \tag{2}$$

of $\mathbf{x}$ and $\mathbf{n} \sim \mathcal{V}$. As before we write $\mathcal{V}_{\mathbf{s}}$ for the resulting distribution of $\mathbf{y}$. This is a generalisation of the obfuscation introduced in Section 2 which can be recovered by choosing $\mathbf{s}$ equal to one on S and zero on S^c. The natural relaxation of the set size $|S|$ is the norm $\|\mathbf{s}\|_1 = \sum_{i=1}^d |s_i|$. As before we define the expected distortion and rewrite it in its bias-variance decomposition

$$D(\mathbf{s}) = \mathbb{E}_{\mathbf{y} \sim \mathcal{V}_{\mathbf{s}}} \left[\frac{1}{2} (\Phi(\mathbf{x}) - \Phi(\mathbf{y}))^2 \right] = \frac{1}{2} (\Phi(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim \mathcal{V}_{\mathbf{s}}} [\Phi(\mathbf{y})])^2 + \frac{1}{2} \mathbb{V}_{\mathbf{y} \sim \mathcal{V}_{\mathbf{s}}} [\Phi(\mathbf{y})], \tag{3}$$

where $\mathbb{V}$ denotes the covariance matrix. The expected distortion is determined by the first and second moment of the output layer distribution. The exact calculation of expectation values and variances for arbitrary functions is in itself already a hard problem. One possibility to overcome this issue is to approximate the expectation by a sample mean. Depending on the dimension d and the distribution $\mathcal{V}$ sampling might be infeasible. Thus, we focus on a second possibility, which takes the specific structure of Φ more into account. From (2) it is straight-forward to obtain the first and second moment

$$\mathbb{E}[\mathbf{y}] = \mathbf{x} \odot \mathbf{s} + \mathbb{E}[\mathbf{n}] \odot (\mathbf{1} - \mathbf{s}) \quad \text{and} \quad \mathbb{V}[\mathbf{y}] = \operatorname{diag}(\mathbf{1} - \mathbf{s}) \mathbb{V}[\mathbf{n}] \operatorname{diag}(\mathbf{1} - \mathbf{s})$$

of the input distribution $\mathcal{V}_{\mathbf{s}}$. It remains to transfer the moments from the input to the output layer. This is discussed in Section 4.3. Instead of the hard constraint $D(\mathbf{s}) \leq \epsilon$ as in (1) we formulate the continuous rate minimisation problem in its Lagrangian formulation

$$\text{minimize} \quad D(\mathbf{s}) + \lambda \|\mathbf{s}\|_1 \quad \text{subject to} \quad \mathbf{s} \in [0, 1]^d \tag{4}$$

with a regularisation parameter $\lambda > 0$. We call this approach to obtaining relevance scores for classifier decisions RDE (Rate-Distortion Explanation). Depending on the activation function, the distortion does not need to be differentiable. However, the ReLU activation is differentiable almost everywhere. As commonly done during the training of neural networks, we simply use (projected) gradient descent to find a stationary point of (4).

4.3 Assumed density filtering

To address the challenge of efficiently approximating the expectation values in (3) we utilise the layered structure of Φ and propagate the distribution of the neuron activations through the network. For this, we use an approximate method, called assumed density filtering (ADF), see for example [13, 3], which has recently been used for ReLU neural networks in the context of uncertainty quantification [8]. In a nutshell, at each layer we assume a Gaussian distribution for the input, transform it according to the layers weights $\mathbf{W}$, biases $\mathbf{b}$, and activation function ϱ, and project the output back to the nearest Gaussian distribution (w.r.t. KL-divergence). This amounts to matching the first two moments of the distribution [13]. We now state the ADF rules for a single network layer. Applying these repeatedly gives us a way to propagate moments through all layers and obtain an explicit expression for the distortion.

Let $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be normally distributed with some mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. An affine linear transformation preserves Gaussianity and acts on the mean and covariance in the well-known way

$$\mathbb{E}[\mathbf{Wz} + \mathbf{b}] = \mathbf{W}\boldsymbol{\mu} + \mathbf{b} \quad \text{and} \quad \mathbb{V}[\mathbf{Wz} + \mathbf{b}] = \mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^*. \tag{5}$$

The ReLU non-linearity ϱ presents a difficulty as it changes a Gaussian distribution into a non-Gaussian one. Let f and F be the probability density and cumulative distribution function of the univariate standard normal distribution, let $\boldsymbol{\sigma}$ be the vector of the diagonal entries of $\boldsymbol{\Sigma}$, and $\boldsymbol{\eta} = \boldsymbol{\mu} \oslash \boldsymbol{\sigma}$. Then as in [8, Equation 10a] we obtain

$$\mathbb{E}[\varrho(\mathbf{z})] = \boldsymbol{\sigma} \odot f(\boldsymbol{\eta}) + \boldsymbol{\mu} \odot F(\boldsymbol{\eta}).$$

Unfortunately the off-diagonal terms of the covariance matrix of $\varrho(\mathbf{z})$ are thought to have no closed form solution [7]. We either make the additional assumption that the network activations within each layer are uncorrelated, which amounts to propagating only the diagonal $\mathbb{V}_{\text{diag}}$ of the covariance matrices through the network and simplifies (5) to

$$\mathbb{V}_{\text{diag}}[\mathbf{Wz} + \mathbf{b}] = (\mathbf{W} \odot \mathbf{W})\,\boldsymbol{\sigma},$$

and results, as also seen in [8, Equation 10b], in

$$\mathbb{V}_{\text{diag}}[\varrho(\mathbf{z})] = \boldsymbol{\mu} \odot \boldsymbol{\sigma} \odot f(\boldsymbol{\eta}) + \left(\boldsymbol{\sigma}^2 + \boldsymbol{\mu}^2\right) \odot F(\boldsymbol{\eta}) - \mathbb{E}[\varrho(\mathbf{z})]^2.$$

Or, we use an approximation for the full covariance matrix

$$\mathbb{V}[\varrho(\mathbf{z})] \approx \mathbf{N}\boldsymbol{\Sigma}\mathbf{N}, \tag{6}$$

with $\mathbf{N} = \operatorname{diag}(F(\boldsymbol{\eta}))$. This ensures positive semi-definiteness and symmetry. Depending on the network size it is usually infeasible to compute the full covariance matrix at each layer. However, if we choose a symmetric low-rank approximation factorisation $\mathbb{V}[\mathbf{y}] \approx \mathbf{Q}\mathbf{Q}^\top$ at the input layer with $\mathbf{Q} \in \mathbb{R}^{d \times r}$ for $r \ll d$ (for example half of a truncated singular value decomposition), then the symmetric update (6) allows us to propagate only one of the factors through the layers. The full covariance is then only recovered at the output layer. This immensely reduces the computational cost and memory requirement.

Altogether, combining the affine linear with the non-linear transformation, tells us how to propagate the first two moments through a ReLU neural network in the ADF framework. We investigate both the *diagonal* as well as the *low-rank* approximation to the covariance matrix in our numerical inquiry.

5 Numerical experiments

We present numerical experiments for interpretable neural networks comparing our proposed method RDE to several state-of-the-art methods. We generate relevance mappings for greyscale images of handwritten digits from the MNIST dataset [10] as well as color images from the STL-10 dataset [5].

As reference distribution $\mathcal{V}$ we consider a Gaussian distribution with mean and variance or low-rank factorisation of the covariance matrix estimated from the training data set as described in [4, 17]. The low-rank factorisation is obtained from a truncated singular value decomposition with rank $r = 30$. During the gradient descent optimisation we use a momentum term with factor 0.85 and determine the step size according to a backtracked Armijo linesearch [15]. We initialise with the constant map $\mathbf{s} = 0.2 \cdot \mathbf{1}_d$ and set the regularisation parameter to $\lambda = 0.5$ for MNIST and $\lambda = 0.05$ for STL-10. The parameters were tuned by grid-search using only one image per data set.

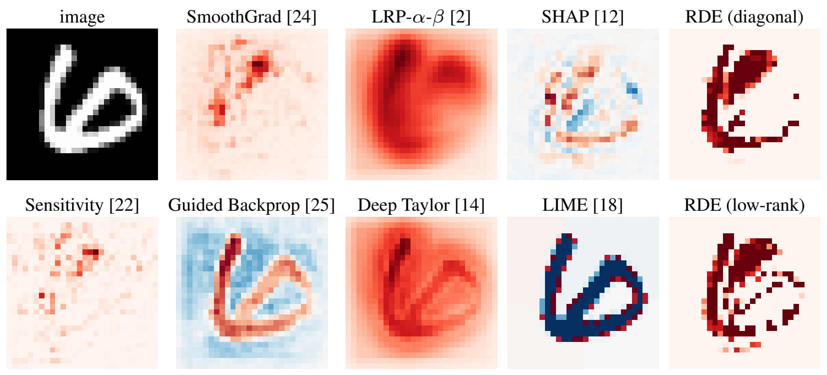

Figure 1: Relevance mappings generated by several methods for an image from the MNIST dataset classified as *digit six* by our network. The colourmap indicates positive relevances as red and negative relevances as blue.

We compare our method to Layer-wise Relevance Propagation (LRP) [2], Deep Taylor decompositions [14], Sensitivity Analysis [22], SmoothGrad [24], Guided Backprop [25], SHAP [12], and LIME [18]. For this we use the Innvestigate[2] [1], SHAP[3], and LIME[4] toolboxes.

Different interpretation approaches produce differently scaled and normalised relevance mappings. Some methods generate non-negative mappings corresponding to the importance *for* the classifier score, whereas other methods also generate negative relevance that can be interpreted as speaking *against* a classifier decision. This has to be dealt with carefully to allow for a fair comparison.

We propose a variant of the *relevance ordering*-based test introduced in [19]. Each relevance mapping induces an ordering of the input signal components by sorting them according to their relevance score (breaking ties randomly). We start with a completely random signal, replace increasingly large parts of it by the original input, and observe the change in the classifier score. This is then averaged over multiple random input samples. A good relevance mapping will lead to a fast convergence to the score of the original signal when the most relevant components are fixed first. In other words the distortion quickly drops to zero. The described process allows to us approximately evaluate the rate-distortion function.

The focus of this paper is the interpretability of neural networks, not on their training so we will keep the description of the training process quite brief.

MNIST experiment We trained a convolutional neural network (three convolution layers each followed by average-pooling and finally two fully-connected layers and softmax output) end-to-end up to a test accuracy of 0.99. The standard training/validation/testing split of the data was used.

The relevance mappings for one example image of the digit six are shown in Figure 1. The mappings are calculated for the pre-softmax score of the class with the highest activation. Both variants of our proposed method generate similar results and highlight an area at the top that distinguishes the digit six from, for example, the digits zero and eight. The relevance-ordering test results are shown in Figure 3 (left). We observe that the expected distortion drops fastest for our proposed method indicating that the most relevant components were correctly identified.

STL-10 experiment We use a VGG-16 network [23] pretrained on the Imagenet dataset and retrain on the STL-10 dataset in three stages. First, we train only the fully-connected layers for 500 epochs, then end-to-end for another 500 epochs, and finally after replacing all max-pooling layers by average-pooling layers again end-to-end for another 500 epochs to a final test accuracy of 0.935.

The relevance mappings for one example image of a dog are shown in Figure 2. As before the mappings are calculated for the pre-softmax score of the class with the highest activation. The difference between both variants of our proposed method is more pronounced here. The low-rank

[2]`https://github.com/albermax/innvestigate`
[3]`https://github.com/slundberg/shap`
[4]`https://github.com/marcotcr/lime`

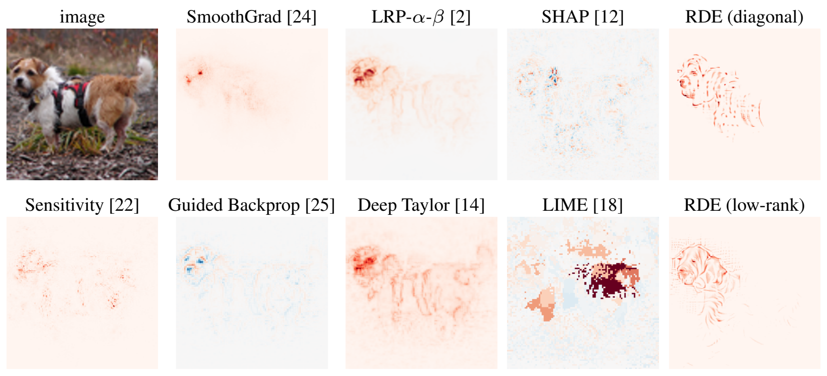

Figure 2: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *dog* by our network. The colourmap indicates positive relevances as red and negative relevances as blue.

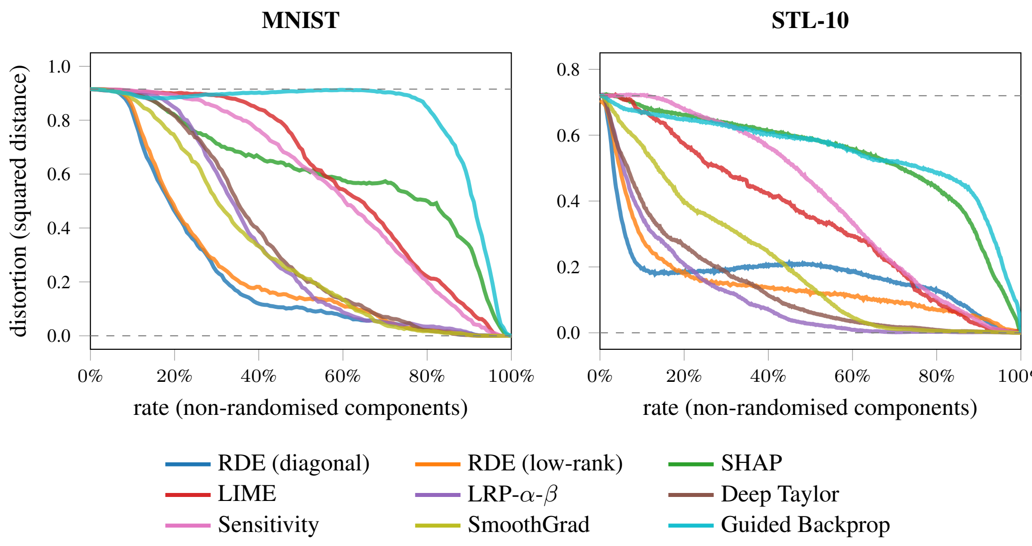

Figure 3: Relevance ordering test results (approximate rate-distortion function) of several methods for the MNIST (left) and STL-10 (right) dataset. An average result over 50 images from the respective test set (5 images per class randomly selected) and 512 (MNIST) and 64 (STL-10) random input samples per image is shown.

variant captures finer details, for example in the dog's face. The relevance-ordering test results are shown in Figure 3 (right). We observe that although our method does not obtain the smallest expected distortions across all rates it has the fastest dropping distortion for low rates indicating that the most relevant components were correctly identified. This is to be expected as our method generates sparse relevance scores highlighting only few of the most relevant components and is not designed for high rates.

6 Conclusion

We introduced a formal rate-distortion framework for explaining classification decisions and analysed the computational complexity of the arising optimisation problem. We saw that in the worst case it is hard to solve and even hard to approximate, which justifies the use of heuristic explanation methods in practical applications. We proposed our method RDE (Rate-Distortion Explanation), which involves a minimisation procedure. We compared it numerically to previous methods and observed that it performs best in the following sense: it captures the smallest set of relevant components, which leads to the steepest curve in the rate-distortion function for small rates.

Acknowledgments

The authors would like to thank Philipp Petersen for several fruitful discussions during the early stage of the project. J. M. and S. W. acknowledge support by DFG-GRK-2260 (BIOQIC). S. H. is grateful for support by CRC/TR 109 “Discretization in Geometry and Dynamics”. G. K. acknowledges partial support by the Bundesministerium für Bildung und Forschung (BMBF) through the “Berliner Zentrum für Machinelles Lernen” (BZML), by the Deutsche Forschungsgemeinschaft (DFG) through Grants CRC 1114 “Scaling Cascades in Complex Systems”, CRC/TR 109 “Discretization in Geometry and Dynamics”, DFG-GRK-2433 (DAEDALUS), DFG-GRK-2260 (BIOQIC), SPP 1798 “Compressed Sensing in Information Processing” (CoSIP), by the Berlin Mathematics Research Centre MATH+, and by the Einstein Foundation Berlin.

References

[1] M. Alber, S. Lapuschki, P. Seegerer, M. Hägele, K. T. Schütt, G. Montavon, W. Samek, K. Müller, S. Dähne, and P. Kindermans. iNNvestigate neural networks! *CoRR*, abs/1808.04260, 2018.

[2] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015.

[3] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI’98, pages 33–42, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[4] T. F. Chan, G. H. Golub, and R. J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. In H. Caussinus, P. Ettinger, and R. Tomassone, editors, *COMPSTAT 1982, 5th Symposium held at Toulouse 1982*, pages 30–41, Heidelberg, 1982. Physica-Verlag HD.

[5] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, Apr 2011. PMLR.

[6] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Learning in graphical models*, pages 75–104. Springer, 1998.

[7] H. Fayed and A. Atiya. An evaluation of the integral of the product of the error function and the normal probability density with application to the bivariate normal integral. *Mathematics of Computation*, 83(285):235–250, 2014.

[8] J. Gast and S. Roth. Lightweight probabilistic deep networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3369–3378, June 2018.

[9] J. Gill. Computational complexity of probabilistic turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[11] M. L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

[12] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

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

[14] G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[15] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[16] J. D. Park. Map complexity results and approximation methods. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 388–396. Morgan Kaufmann Publishers Inc., 2002.

[17] R. Řehůřek. *Scalability of Semantic Analysis in Natural Language Processing*. PhD thesis, Masaryk University, Faculty of Informatics, Brno, Czech Republic, May 2011.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin. "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, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[19] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller. 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.

[20] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[21] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. *CoRR*, abs/1704.02685, 2017.

[22] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[24] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017.

[25] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (Workshop)*, 2015.

[26] S. Waeldchen, J. Macdonald, S. Hauch, and G. Kutyniok. The computational complexity of understanding network decisions. *arXiv e-prints*, page arXiv:1905.09163, May 2019.

[27] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.