Methods for interpreting and understanding deep neural networks

Seminar Report: Explainable Machine Learning SS18

Philipp Wimmer

July 9, 2018

Contents

1 Motivation

Our goal in the seminar ”Explainable Machine Learning” is to gain insight into how the state-of-the-art machine learning models work. Prior to this seminar, the models were defined on an algorithmic level. However, the new goal is to get a deep qualitative understanding of how they work. Enabling us to gage how well we can trust them. The models were seen as a black box. The goal is to open this box.
The paper that was chosen is ”Methods for interpreting and understanding deep neural networks” from Montavon et. al.[3]. Deep Neural networks work exceedingly well in many machine learning tasks and for example enable huge advancements in the field of computer vision. The universal approximation theorem gives us mathematical proof that these networks can (in theory) learn any concept. Training these models via gradient descent is also rather straight forward. Also the success can be seen by the fact that nearly every top spot in the many machine learning challenges is occupied by a deep learning algorithm. But this performance comes at the price of higher model complexity. Whereas the shallow models used earlier (like logistic regression) had clear limitations and were easy to understand, this new class of universal approximators is way harder to verify. The most used way of verifying DNNs is to measure the performance on a test data set. This is not enough information to trust the decisions made by these deep models. We do not know how the decision was made and if the model learned the right concept. The model may exhibit failure modes that are simply not found by testing on the test set. The paper introduces two fundamental approaches to solve this problem: Interpretation and Explanation. Both methods are for better post-hoc interpretability, thus enabling the exploration of already existing and even pretrained DNNs.

2 Theoretical Background

2.1 Bayes' Theorem

Combining the two rules of probability and using the symmetry property of joint probability

$$\textbf{sum rule } p(A) = \sum_B p(A,B) \tag{2.1}$$

$$\textbf{product rule } p(A,B) = p(B|A)p(A) \tag{2.2}$$

$$\textbf{symmetry } p(A,B) = p(B,A) \tag{2.3}$$

one arrives at:

$$\textbf{Bayes' Theorem } p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)} \tag{2.4}$$

$$\tag{2.5}$$

Applied to machine learning models, where $\mathbf{x}$ represents the data and w_c is the learned class, one gets:

$$p(w_c|\mathbf{x}) = \frac{p(\mathbf{x}|w_c) \cdot p(w_c)}{p(\mathbf{x})} \tag{2.6}$$

This theorem plays a fundamental role in the statistical interpretation of machine learning models. Models can be divided into two different groups: discriminative and generative models. Discriminative models approximate the decision boundary between the classes. They model the left side of the equation $p(w_c|\mathbf{x})$ called the posterior probability. Generative models learn the right side of the equation. They effectively learn the joint probability $p(w_c, \mathbf{x})$, thus enabling the generation of new data by sampling from this distribution.

2.2 DNN

Deep neural networks (DNNs) are a special kind of artificial neural networks with a deep hierarchical structure. They are able to learn complex relationships between their input and output parameters. They fall under the category of discriminative models. The activations of the individual neurons are calculated by:

$$a_k = \sigma\left(\sum_j a_j w_{jk} + b_k\right) \tag{2.7}$$

σ is a nonlinearity and w_{jk} is the matrix of the learned network weights. Provided that the output linearity used is the softmax function $O_j = \frac{e^{a_j}}{\sum_k e^{a_k}}$ and a suitable loss function is used (for example cross entropy loss), the output of the network approximates the posterior probability $p(w_c|\mathbf{x})$. Multilayer networks (with at least one hidden layer) can approximate any continuous function on compact subsets of $\mathbf{R}^n$ if their capacity is high enough. See proof in [2]. The problem with these models is that despite their high performance, they do not inspire trust in their results. This is due to the fact that there is no obvious way to interpret the (possibly very large) learned weight vector. The performance is usually measured by cross validation but this does not rule out failures when working on real life data.

2.3 Gaussian restricted Boltzmann machines

Gaussian restricted Boltzmann machines introduced by Salakhutdinov et. al [4] fall under the category of generative models. They are a class of two layer undirected graphical models and are able to represent complex distributions. Their log-probability function can be written as:

$$\log p(\mathbf{x}) = \sum_j f_j(\mathbf{x}) - \lambda||\mathbf{x}||^2 + const \tag{2.8}$$

where

$$f_j(\mathbf{x}) = log(1 + \exp(\mathbf{w}_j^\top \mathbf{x}) + b_j) \tag{2.9}$$

are factors learned from the training data.

2.4 Generative Adverserial Autoencoders

Generative Adverserial Autoencoders (GANs) as introduced by Goodfellow et. al. [1] fall under the category of generative models. They consist of two different models that are trained in an adversarial training procedure: A generative model G captures the data distribution and a discriminator D tries to differentiate between data generated by G and the training data. G generates its output from a random noise vector $\mathbf{z}$. Thus, they play a minimax game with the value function $V(G, D)$.

$$\min_G \max_D = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\log(1 - D(G(\mathbf{z})))\right] \tag{2.10}$$

3 Interpretation

The authors define an interpretation as:

> *(...) the mapping of an abstract concept (e.g. a predicted class) into a domain that the human can make sense of.*

In the context of object classification, the interpretable domain is the space of all possible images. In the concrete example provided in the paper, the abstract concept is the range of digits from 0 to 9. The result of an interpretation is the construction of a prototype image for a given class. In the perfect case, these prototype images would be indistinguishable from real images and appropriately represent the concept one wanted the DNN to learn.
The task of interpretation is an interesting one because one tries to generate data with a discriminative model. This poses problems because the posterior probability, that the model approximates, is not enough to generate data.
In the following the technique of Activation maximization is presented as a tool to construct these prototypes. This simple approach will later be improved upon to produce more satisfying results.

3.1 Activation maximization

The goal of Activation maximization (AM) is to maximize the activation of the output neurons of the DNN. The idea is to present an initial random image only consisting of noise to the DNN and calculate the response. Then, the response is maximized via gradient ascent. This is similar to the training phase only that the objective is changed. Instead of the weights the individual pixels of the image get modified. This corresponds to maximizing the following expression:

$$\max_{\mathbf{x}} \log p(w_c|\mathbf{x}) - \lambda||\mathbf{x}||^2 \tag{3.1}$$

The class probability $p(w_c|\mathbf{x})$ is encoded in the output neurons and the ℓ^2-Regularizer $\lambda||\mathbf{x}||^2$ is needed because there are many local minima. Without this regularizer the maximization would not be stable. Subsequently, inputs close to the origin are favored. This preference to the origin creates mostly gray sparse images with structure only in certain parts of the image region. These prototypes do not look natural and thus do not encourage a high trust that the DNN learned the right concept.

3.2 Activation maximization with an expert

The solution to the problem of unnatural images is to introduce a more sophisticated regularizing term. The ℓ^2-Regularizer is replaced with a model of the data.

$$\max_{\mathbf{x}} \log\, p(w_c|\mathbf{x}) + \log\, p(\mathbf{x}) \tag{3.2}$$

On first sight, the difference is not fundamental. By applying Bayes' theorem, however, one can see that maximizing the expression is equivalent to maximizing the class-conditioned data density $p(\mathbf{x}|w_c)$.

$$\begin{aligned} p(w_c|\mathbf{x}) &= \frac{p(\mathbf{x}|w_c) \cdot p(w_c)}{p(\mathbf{x})} \\ \log\, p(w_c|\mathbf{x}) &= \log\, p(\mathbf{x}|w_c) + \log\, p(w_c) - \log\, p(\mathbf{x}) \\ \log\, p(\mathbf{x}|w_c) &= \log\, p(w_c|\mathbf{x}) + \log\, p(\mathbf{x}) - \log\, p(w_c) \\ \log\, p(\mathbf{x}|w_c) &= \log\, p(w_c|\mathbf{x}) + \log\, p(\mathbf{x}) + const \end{aligned}$$

Provided that a suitable $p(\mathbf{x})$ was used, this quantity actually represents a good prototype. The found prototype $\mathbf{x}^*$ then represents the most likely input for a given concept w_c.

This additional term enables the creation of data through a discriminative model. But this data density $p(\mathbf{x})$ has to be learned by a (possibly very complex) generative model and may be difficult to maximize. In practice, Gaußian RBMs are used for this purpose. Additionally, the right choice of expert is very important. Even though the perfect case would be the real data density, this goal is often not achievable. While an overfitted expert may produce realistic looking prototypes, it may also hide interesting failure modes of the model. In this case the prototypes only fit the expert and the role of the model diminishes. For verification, an underfitted expert is preferable because it favors natural images and does not hide failure modes.

3.3 Activation maximization in Code Space

Learning the data density can be quite complex and is often difficult to maximize. In this step, a Generative Adverserial Autoencoder is used that learns the data density implicitly. Instead of maximizing in the interpretable domain, the maximization takes place in an abstract code space $\mathcal{Z}$.

$$\max_{\mathbf{z}\in\mathcal{Z}} \log\, p(w_c|g(\mathbf{z})) - \lambda\, ||\mathbf{z}||^2 \tag{3.3}$$

The expression that gets maximized shares a striking resemblance to eq. (3.1) but in fact it is mathematically much closer to eq. (3.2). This is because the code space is gaußian distributed.

$$p(\mathbf{z}) \propto \exp(-x^2) \tag{3.4}$$

$$\log\, p(\mathbf{z}) \propto -x^2 \tag{3.5}$$

Thus maximizing eq. (3.3) is equivalent to:

$$\max_{\mathbf{z}\in\mathcal{Z}} \log p(g(\mathbf{z})|w_c) = \max_{\mathbf{z}\in\mathcal{Z}} \log p(w_c|g(\mathbf{z})) + \log p(\mathbf{z}) \tag{3.6}$$

Again, the resulting prototype $\mathbf{x}^*$ is the most likely point for the concept w_c.

3.4 Experimental Results

The authors tested the implementations on the MNIST-dataset. One can see that the version with the simple ℓ^2-Regularizer performs badly and produces unsharp images. The quality of the version with the expert and the version with the decoder perform equally well.

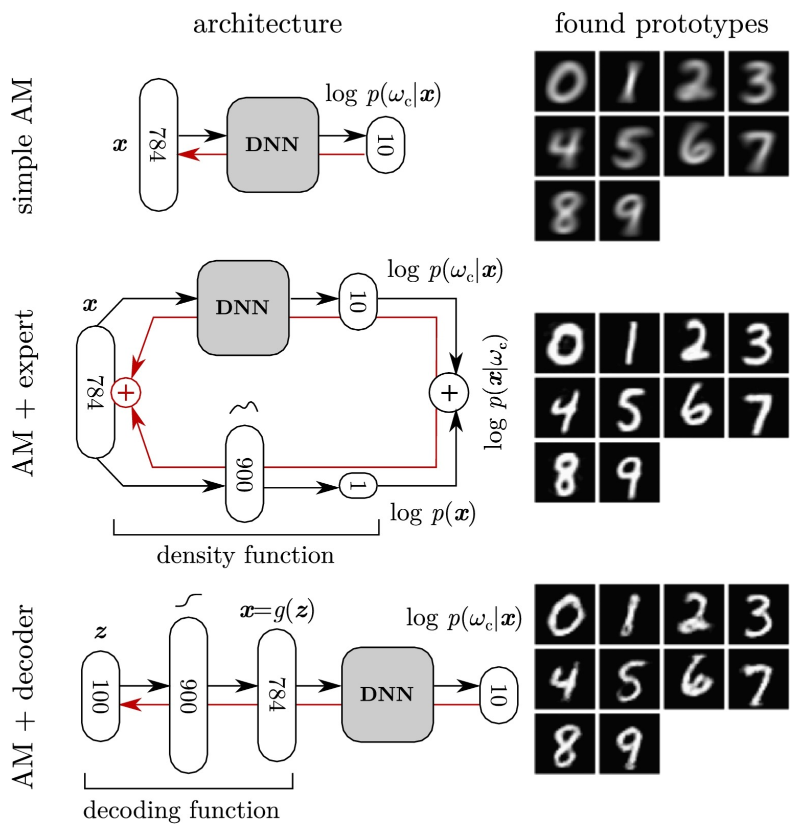

Figure 3.1: The architecture of activation maximization: One can see that the objective for the simple AM $\log p(w_c|\mathbf{x})$ differs from the objective of the two other architectures $\log p(\mathbf{x}|w_c)$. In the first case the class probability (posterior) gets maximized and in the second and third the class conditioned data density (likelihood) gets maximized, rendering way better results.

4 Explanation

The authors defined an explanation as:

> *(...) the collection of features of the interpretable domain, that have contributed for a given example to produce a decision (e.g. classification or regression).*

In the context of image classification, an explanation is the collection of all pixels that led the models to its decision. This can be visualized as a heatmap. More precisely, it is a visualization of how important each pixel was in order to arrive at the decision. This can, for example, lead to the discovery of anomalies in the model. The test data may be classified correctly but the DNN exploited an anomaly in the training data (always the same background for the same classes). Consequently, it may exhibit bad generalization performance.

4.1 Sensitivity Analysis

The goal of sensitivity analysis is to assign a local relevance score to each individual pixel. Relevance is usually defined as a measurement of local slope of the models decision function f.

$$R_i(\mathbf{x}) = \left(\frac{\partial f}{\partial x_i}\right)^2 \tag{4.1}$$

The problem with this simple definition is that it is only a measurement of local variance. The obtained scores are not the answer to the question "How much did pixel i contribute to the decision?" Instead one asks "How much did pixel i make the data belong more or less to the predicted class?"

4.2 Simple Taylor decomposition

One way to produce absolute relevance scores is the simple Taylor decomposition. Now the relevance scores are defined to be a decomposition of the relevance scores. The scores R_i are defined in terms of the Taylor expansion.

$$f(\mathbf{x}) = \sum_{i=1}^{d} R_i(\mathbf{x}) + \mathcal{O}(\mathbf{x}\mathbf{x}^\top) \tag{4.2}$$

$$R_i(\mathbf{x}) = \left.\frac{\partial f}{\partial x_i}\right|_{\mathbf{x}=\tilde{\mathbf{x}}} \cdot (x_i - \tilde{x}_i) \tag{4.3}$$

The decomposition in these scores is not exact because the higher order terms $\mathcal{O}(\mathbf{x}\mathbf{x}^\top)$ are neglected. However, piecewise functions that satisfy the property

$$f(t\,\mathbf{x}) = t\,f(\mathbf{x}), t \geq 0 \tag{4.4}$$

do not exhibit this limitation. There $f(\mathbf{x})$ can be decomposed exactly by

$$f(\mathbf{x}) = \sum_{i=1}^{d} R_i(\mathbf{x}) \tag{4.5}$$

$$R_i(\mathbf{x}) = \frac{\partial f}{\partial x_i} \cdot x_i \tag{4.6}$$

The obtained relevance score can be interpreted as the product of local sensitivity $\frac{\partial f}{\partial x_i}$ and saliency x_i. Thus, a high relevance score does not only mean that the model reacts to it, but also that it is present in the data.

4.3 Layer-wise relevance propagation

An alternative to decomposition models are backward propagation techniques. In layer-wise relevance propagation (LRP) Relevance is defined to be a quantity that is distributed and flowing backwards through the network. Relevance has to satisfy the conservation property

$$\sum_j R_{j \leftarrow k} = R_k \tag{4.7}$$

where $R_{j \leftarrow k}$ is the relevance flowing from a neuron in layer k to a higher level j. Local conservation of relevance does also imply

$$R_j = \sum_k R_{j \leftarrow k} \tag{4.8}$$

Combining these two equations also ensures the total conservation of Relevance.

$$\sum_j R_j = \sum_j \sum_k R_{j \leftarrow k} = \sum_k \sum_j R_{j \leftarrow k} = \sum_k R_k = f(\mathbf{x}) \tag{4.9}$$

One propagation rule satisfying these conditions is the $\alpha\beta$-rule

$$R_j = \sum_k \left(\alpha \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} - \beta \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-} \right) R_k \tag{4.10}$$

where the two hyperparameters α and β have to satisfy the constraints $\alpha - \beta = 1$ and $\beta \geq 0$. These define the relative importance of the positive and negative parts. The rule can be rewritten to

$$R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k^\wedge + \sum_k \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-} R_k^\vee \tag{4.11}$$

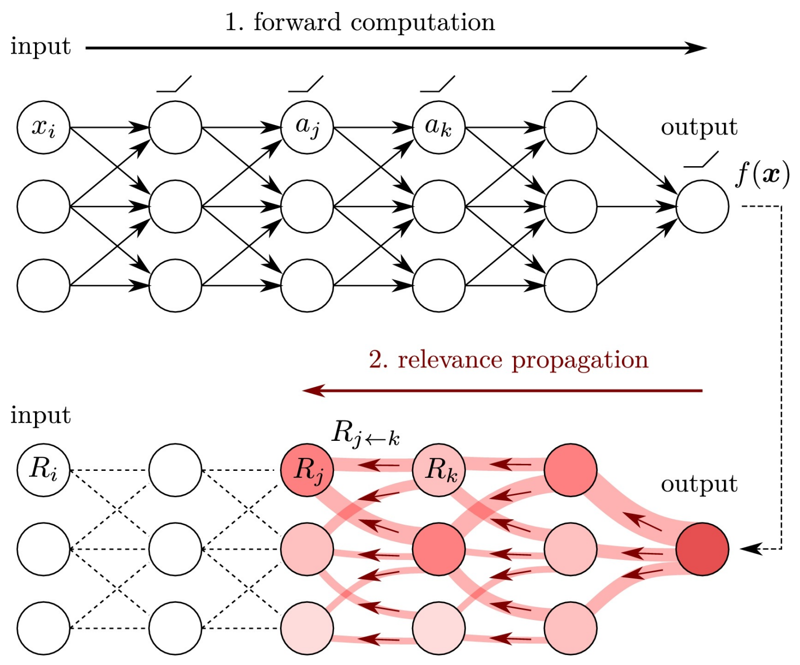

Figure 4.1: Visualization of the backwards flow of Relevance through a deep neural network.

with

$$R_k^{\wedge} = \alpha R_k \tag{4.12}$$

representing Relevance and

$$R_k^{\vee} = -\beta R_k \tag{4.13}$$

representing counter-Relevance. There are special variants of this rule for the speciality layers used in modern neural networks. The authors also provided proof that LRP with $\alpha = 1$ and $\beta = 0$ is equivalent to layerwise application of the simple Taylor decomposition.

4.4 Experimental Results

The authors of the paper tested the implementations on the MNIST-Dataset. One can see that the simple Taylor decomposition does not produce satisfying results. It does produce a large proportion of the pixel to go against class evidence. The authors argue that this is due to the fact that root point $\tilde{\mathbf{x}}$ used in the Taylor expansion is too dissimilar to the actual data $\mathbf{x}$. Thus, it does not contextualize the explanation appropriately.

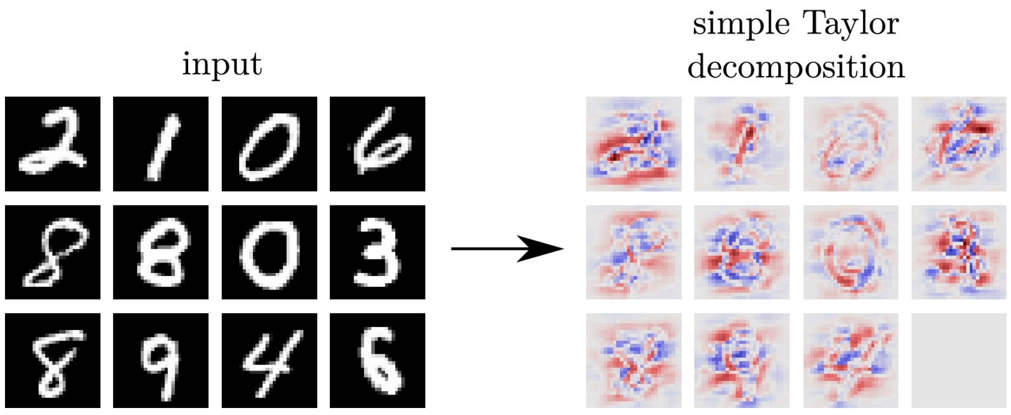

Figure 4.2: Heatmaps obtained by simple Taylor Decomposition. Red corresponds to Relevance and blue to counter-Relevance. Too much counter-Relevance is distributed.

LRP does produce way more satisfying results. The produced heatmaps appropriately map to the regions of the images that would be labeled as relevant by a human observer. The authors stress that the best values for the hyperparameters have to be chosen experimentally for each model. For the concrete model used in the paper $\alpha = 3$ and $\beta = 2$ produce unstable results with too much counter-Relevance being accounted.

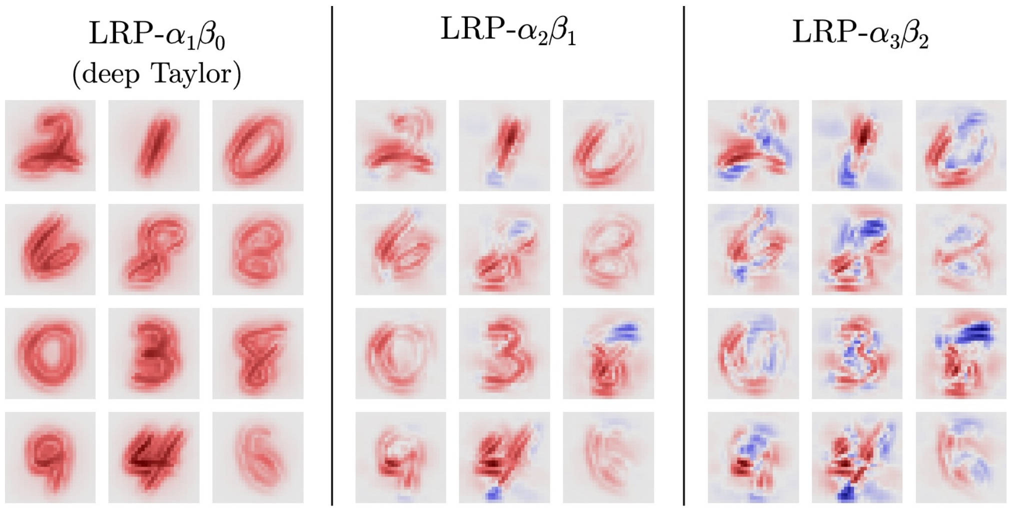

Figure 4.3: Heatmaps obtained by LRP with varying hyperparameters. Red corresponds to Relevance and blue to counter-Relevance. For $\beta = 0$ no counter-Relevance is distributed and for $\beta > 1$ the results get unstable.

5 Conclusion

The authors of the paper provided two different approaches in order to enable better understanding and verification of DNNs. These approaches can be applied to already trained DNNs and there is no need to make any modifications. Moreover, they only demand a small investment to implement. Thus, they should be essential tools for anyone working with DNNs. Finally, the construction of prototypes enables the investigation of the concept that the model has learned and if it is identical to the one humans wanted it to learn. The heatmaps, produced as Explanations, nicely visualize the decision process. They also make it possible to examine how sensitive the model is to the context. The authors did manage to present two ways to enlarge the trust in the decision made by DNNs. Thus, they succeeded at narrowing the interpretability gap towards simpler models like logistic regression.

Bibliography

[1] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[2] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[3] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. “Methods for interpreting and understanding deep neural networks”. In: *Digital Signal Processing* (2017).

[4] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. “Restricted Boltzmann Machines for Collaborative Filtering”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML ’07. Corvalis, Oregon, USA: ACM, 2007, pp. 791–798. ISBN: 978-1-59593-793-3. DOI: `10.1145/1273496.1273596`. URL: `http://doi.acm.org/10.1145/1273496.1273596`.