Explainable neural network

Li, Heng
HPCA-AI, CS department, Tsinghua University
Room 202, 8th block, East main building, Tsinghua university
liheng19@mails.tsinghua.edu.cn

Abstract

This paper provides a survey to the problem of the explainable deep neural network model and explaining its predictions. It is based on a tutorial given at High Level Machine Learning Course 2019. As a tutorial paper, the set of methods covered here is not exhaustive, but sufficiently representative to discuss a number of questions in eXplainable Neural Network (XNN), technical challenges, and possible applications.

1 Introduction

Explainable AI (XAI) refers to methods and techniques in the application of artificial intelligence technology (AI) such that the results of the solution can be understood by human experts. It contrasts with the concept of the "black box" in machine learning where even their designers cannot explain why the AI arrived at a specific decision.

As one of the most important and rapidly growing branch of AI, deep neural networks have become an indispensable tool for a wide range of applications such as image classification, speech recognition, or natural language processing. These techniques have achieved extremely high predictive accuracy, in many cases, on par with human performance.

In practice, it is also essential to verify for a given task, that the high measured accuracy results from the use of a proper problem representation, and not from the exploitation of artifacts in the data. Techniques for interpreting and understanding what the model has learned have therefore become a key ingredient of a robust validation procedure. Interpretability is especially important in applications such as medicine or self-driving cars, where the reliance of the model on the correct features must be guaranteed. Machine learning models, especially for deep neural network, remain mostly black boxes [17] unable to explain the reasons behind their predictions or recommendations, thus eroding users trust.

Techniques of interpretation are also becoming increasingly popular as a tool for exploration and analysis in the sciences. In combination with deep nonlinear machine learning models, they have been able to extract new insights from complex physical, chemical, or biological systems.

This tutorial gives an overview of techniques for explainable deep neural networks (XNN). It starts by discussing the problem of interpreting modeled concepts (e.g. predicted classes), and then moves to the problem of explaining individual decisions made by the model.The tutorial abstracts from the exact neural network structure and domain of application, in order to focus on the more conceptual aspects that underlie the success of these techniques in practical applications.

In spite of the practical successes, one should keep in mind that explainable deep networks remains a young and emerging field of research. There are currently numerous coexisting approaches to explanation. This tutorial gives a snapshot of the field at present time and it is naturally somewhat biased towards the authors view; as such we hope that it provides useful information to the reader.

Preprint. Under review.

2 Preliminaries

Techniques of explaining have been applied to a wide range of practical problems, and various meanings have been attached to terms such as understanding, interpreting, or explaining. See [9] for a discussion. As a first step, it can be useful to clarify the meaning we assign to these words in this tutorial, as well as the type of techniques that are covered.

Generally, if we could exchange the ideas with each other, We should not challenge the explainable of the decision of a human beings even we have no idea about the structure of his/her brain. In a same way, we will focus in this tutorial on post-hoc interpretability, i.e. a trained model is given and our goal is to understand what the model predicts (e.g. categories) in terms what is readily interpretable (e.g. the input variables).

Throughout this tutorial, we will make a distinction between interpretation and explanation, by defining these words as follows [11].

Interpretation

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

Examples of domains that are interpretable are images (arrays of pixels), or texts (sequences of words). A human can look at them and read them respectively. Examples of domains that are not interpretable are abstract vector spaces (e.g. Auto-encoder), or domains composed of undocumented input features (e.g. sequences with unknown words or symbols).

Explanation

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

The features that form the explanation can be supplemented by relevance scores indicating to what extent each feature contributes. Practically, the explanation will be a real-valued vector of same size as the input, where relevant features are given positive scores, and irrelevant features are set to zero.

An explanation can be, for example, a heatmap highlighting which pixels of the input image most strongly support the classification decision [19], [2]. In natural language processing, explanations can take the form of highlighted text[8].

For a real-world prediction such like movie rating and web page popularity, since the evidence on the input data is much more complex which is not easy to compute the evidience and also is not straight-forward to explanation, we try to introduce prior knowledge (variational inference) to draw the semantic or concise representation of data [4], [3].

Causality

Causality is efficacy, by which one process or state, a cause, contributes to the production of another process or state.The cause is partly responsible for the effect, and the effect is partly dependent on the cause.

The aim of standard statistical analysis (such like interpretation and explanation) is to assess parameters of a distribution from samples drawn of that distribution. With the help of such parameters, associations among variables can be inferred, which permits the researcher to estimate probabilities of past and future events and update those probabilities in light of new information. These tasks are managed well by standard statistical analysis so long as experimental conditions remain the same. Causal analysis goes one step further; its aim is to infer probabilities under conditions that are changing, for example, changes induced by treatments or external interventions.

Base on the Structural Causal Model (SCM) developed in (Pearl, 1995, [16]), casual inference will help us to find out the casual effect of the input on the output of the given DNN.

3 Interpreting a DNN model with Bayesian reference

This section focuses on the problem of interpreting a concept learned by a deep neural network (DNN). A DNN is a collection of neurons organized in a sequence of multiple layers, where neurons

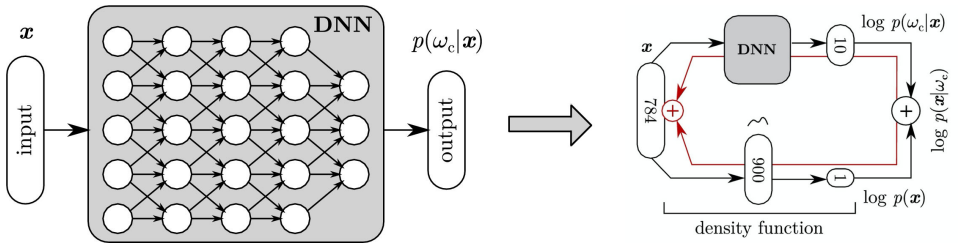

Figure 1: Example of a discriminative neural network composed of many interconnected neurons, and that assigns to the input x a probability of being associated to a certain concept w_c which can implement $P(w_c|x)$ inference. Base on attaching a $P(x)$ model, the given discriminative DNN can enhanced into a generative model to implement $P(x|w_c)$ inference.

receive as input the neuron activation from the previous layer, and perform a simple computation (e.g. a weighted sum of the input followed by a nonlinear activation). The neurons of the network jointly implement a complex nonlinear mapping from the input to the output. This mapping is learned from the data by adapting the weights of each neuron using a technique called error back-propagation. An example of a neural network is shown in Fig. 1.

The concept that must be interpreted is usually represented by a neuron in the top layer. Top-layer neurons are abstract (i.e. we cannot look at them), on the other hand, the input domain of the DNN (e.g. image or text) is usually interpretable. We describe below how to build a prototype in the input domain that is interpretable and representative of the abstract learned concept. Building the prototype can be formulated within the activation maximization framework.

Activation maximization is an analysis framework that searches for an input pattern that produces a maximum model response for a quantity of interest [6], [19].

Consider a DNN classifier mapping data points x to a set of classes $\{w_c\}_c$. The output neurons encode the modeled class probabilities $P(w_c|x)$. Base on Bayesian inference, a prototype x^* representative of the class $\hat{w}_c$ can be found by optimizing:

$$x^* = \underset{x}{\text{argmax}} \log p(\hat{w}_c, x) = \underset{x}{\text{argmax}} \log p(\hat{w}_c|x)p(x) \tag{1}$$

If the given DNN is a generative model which defined $P(w_c, x)$, it shall be very easy to inference x^* from $\hat{w}_c$. But if the given DNN is a discriminative model which only carry $P(w_c|x)$, we shall attach $P(x)$ on this model which is shown in Fig. 1.

3.1 Simple AM

We can start from taking the prior probability $X \sim N(0, \sigma^2)$ and the rightmost term of the objective is an l_2-norm regularizerhere which indicates that the model take consideration with [19]. The class probabilities modeled by the DNN are functions with a gradient. This allows for optimizing the objective by gradient ascent.

$$x^* = \underset{x}{\text{argmax}} \log p(\hat{w}_c, x) = \underset{x}{\text{argmax}} \log p(\hat{w}_c|x) - \lambda||x||^2$$

3.2 Improving AM with an expert

In order to obtain more meaningful prototypes, the l_2-regularizer can be replaced by a more sophisticated one [12] called expert. The expert can be, for example, a model $p(x)$ of the data. This leads to the new optimization problem.

The prototype x^* obtained by solving this optimization problem will simultaneously produce strong class response and resemble the data. By application of the Bayes' rule, the newly defined objective can be identified, up to modeling errors and a constant term, as the class-conditioned data density $P(x|w_c)$. The learned prototype will thus correspond to the most likely input x^* for class $\hat{w}_c$.

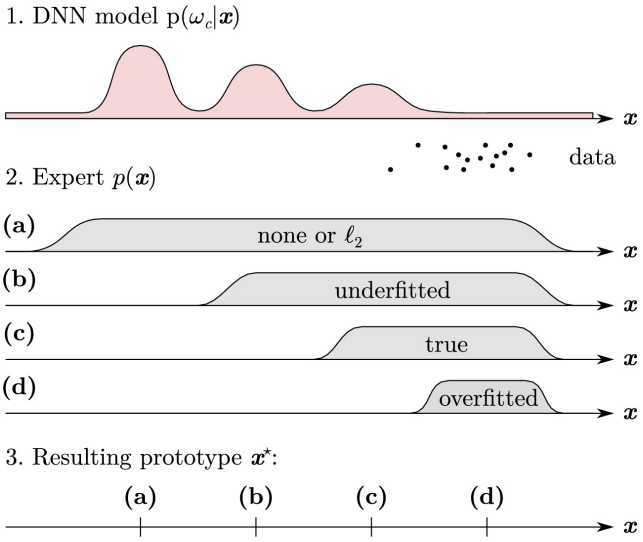

Figure 2: Cartoon illustrating how the choice of expert $p(x)$ affects the prototype x^* found by AM. The horizontal axis represents the input domain.

A possible choice for the expert is the Gaussian RBM. It can represent complex distributions and has a gradient in the input domain. Its log-probability function can be written as:

$$\log p(x) = \sum_j f_j(x) - ||x||^2 + \text{cst.},$$

where the terms $f_j(x) = \log(1 + \exp(w_j^T x + b_j))$ are learned from the data, and come in superposition to the original l_2-norm regularizer. When interpreting concepts such as natural images classes, more complex density models such as convolutional RBM/DBMs [7], or pixel-RNNs [15] can be used instead. In practice, the choice of the expert $p(x)$ plays an important role in determining the appearance of the resulting prototype. The dependence of the prototype on the choice of expert is illustrated in Fig. 2.

On one extreme a coarse expert (a) reduces the optimization problem to the maximization of the class probability function $p(w_c|x)$. On the other extreme an overfitted expert (d) essentially reduces the optimization problem to the maximization of the expert $p(x)$ itself.

When using AM for the purpose of validating a DNN model, an overfitted expert (d) must be especially avoided, as the latter could hide interesting failure modes of the DNN model. A slightly underfitted expert (b), e.g. that simply favors images with natural colors, can therefore be sufficient. On the other hand, when using AM to gain knowledge on a concept w_c correctly predicted by the DNN, the focus should be to prevent underfitting. Indeed, an underfitted expert (b) would expose optima of $p(w_c|x)$ potentially distant from the data, and therefore, the prototype x^* would not be truly representative of w_c. Hence, it is important in that case to learn a density model as close as possible to the true data distribution (c).

3.3 Performing AM in code space

In certain applications, data density models $p(x)$ can be hard to learn up to high accuracy, or very complex such that maximizing them becomes difficult, or x is not interpretive, too. We shall involve latent variable which is interpretive and simple (yes the latent variables are sully exist as the world is really simple comparing with the complex data.). Some optional models used to involve latent variable are listed in Fig. 3.

For example, an alternative class of unsupervised models are generative models. They do not provide the density function directly, but are able to sample from it, usually via the following two steps:

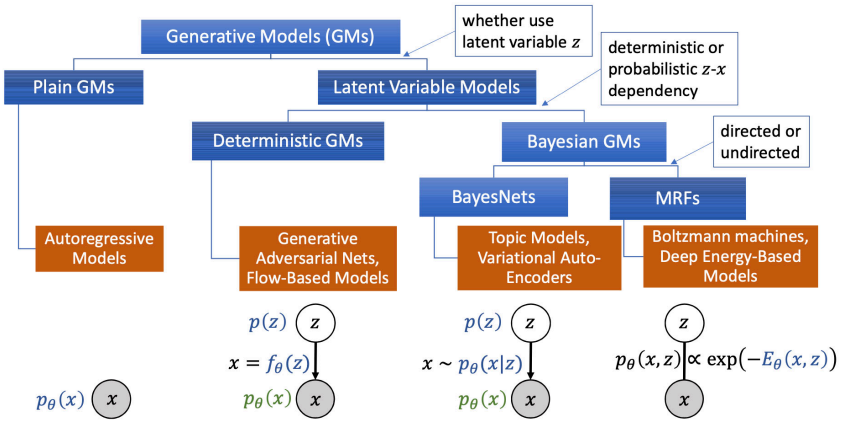

Figure 3: Generative Model: Taxonomy)

1. Sample from a simple distribution $q(z) \sim N(0, I)$ (or other prior distribution) defined in some abstract code space $\mathcal{I}$.

2. Apply to the sample a decoding function $g : \mathcal{I} \to \mathcal{H}$, that maps it back to the original input domain.

One such model is the generative adversarial network. It learns a decoding function g such that the generated data distribution is as hard as possible to discriminate from the true data distribution. The decoding function g is learned in competition with a discriminant between the generated and the true distributions. The decoding function and the discriminant are typically chosen to be multilayer neural networks.

Nguyen et al.[13] proposed to build a prototype for w_c by incorporating such generative model in the activation maximization framework. The optimization problem is redefined as:

$$x^* = \underset{x}{\operatorname{argmax}} \log p(\hat{w}_c, x) = \max_{z \in \mathcal{I}} \log p(w_c | g(z)) - \lambda ||z||^2,$$

where the first term is a composition of the newly introduced decoder and the original classifier, and where the second term is an l_2-norm regularizer in the code space. Once a solution z^* to the optimization problem is found, the prototype for w_c is obtained by decoding the solution, that is, $x^* = g(z^*)$.

When the code distribution $q(z)$ is chosen to be a normal distribution, the l_2 penalty $-||z||^2$ becomes equivalent (up to a scaling factor and a constant) to $\log q(z)$, and can therefore be understood as favoring codes with high probability. However, as high probability codes do not necessarily map to high density regions in the input space, the maximization in code space described in this section will only approximately optimize the desired quantity $\log p(x|w_c)$.

To illustrate the qualitative differences between the methods of Sections 3.1 to 3.3, we consider the problem of interpreting MNIST classes as modeled by a three-layer DNN. We consider for this task (1) a simple l_2-norm regularizer $||x - \overline{x}||^2$ where $\overline{x}$ denotes the data mean for w_c, (2) a Gaussian RBM expert $p(x)$, and (3) a generative model with a two-layer decoding function, and the l_2-norm regularizer $||z - \overline{z}||^2$ where $\overline{z}$ denotes the code mean for w_c. Corresponding architectures and found prototypes are shown in Fig. 4. Each prototype is classified with full certainty by the DNN. However, only with an expert or a decoding function, the prototypes become sharp and realistic-looking.

3.4 From global to local analysis

When considering complex machine learning problems, probability functions $p(w_c|x)$ and $p(x)$ might be multimodal or strongly elongated, so that no single prototype x^* fully represents the modeled concept w_c. The issue of multimodality is raised by Nguyen et al. [14], who demonstrate in the context of image classification, the benefit of interpreting a class w_c using multiple local prototypes instead of a single global one.

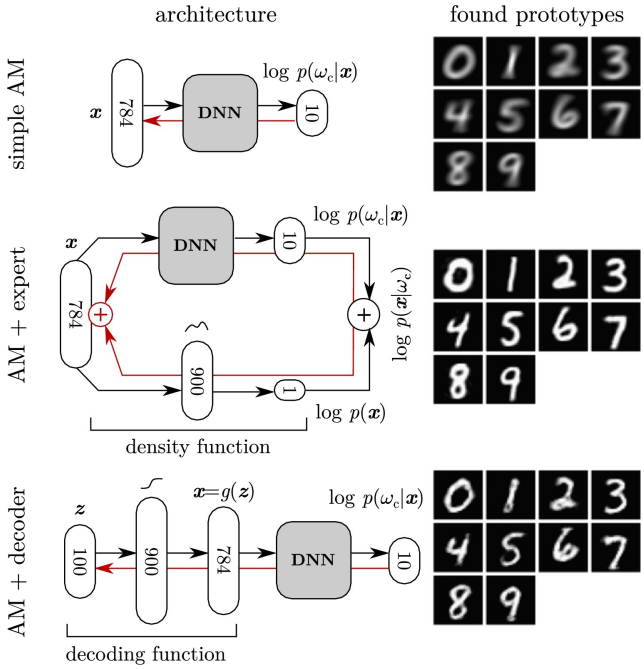

Figure 4: Architectures supporting AM procedures and found prototypes. Black arrows indicate the forward path and red arrows indicate the reverse path for gradient computation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Producing an exhaustive description of the modeled concept w_c is however not always necessary. One might instead focus on a particular region of the input space. For example, biomedical data is best analyzed conditioned on a certain development stage of a medical condition, or in relation to a given subject or organ.

An expedient way of introducing locality into the analysis would be to add a localization term $||x - x_0||^2$ to the AM objective, where x_0 is a reference point. The parameter controls the amount of localization. As this parameter increases, the question what is a good prototype of w_c? becomes however insubstantial, as the prototype x^* converges to x_0 and thus looses its information content.

Instead, when trying to interpret the concept w_c locally, a more relevant question to ask is what features of x make it representative of the concept w_c?. This question gives rise to a second type of analysis, explaining DNN decisions, that will be the focus of the rest of this tutorial.

4 Explaining DNN decisions

In this section, we ask for a given data point x, what makes it representative of a certain concept w_c encoded at the output of the deep neural network (DNN). The output neuron that encodes this concept can be described as a function $f(x)$ of the input.

A common approach to explanation is to view the data point x as a collection of features $(x_i)_{i=1}^d$, and to assign to each of these, a score R_i determining how relevant the feature x_i is for explaining $f(x)$. An example is given in Fig. 5.

In this example, an image is presented to the DNN, that finds some evidence for class boat. The prediction is then mapped back to the input domain. The explanation takes the form of a heatmap, where pixels with a high associated relevance score are shown in red. In this example, the explanation procedure rightfully assigns relevance to the pixels representing actual boats in the image, and ignores most of the pixels in the background. In the next sections, we present several candidate methods for producing these relevance scores.

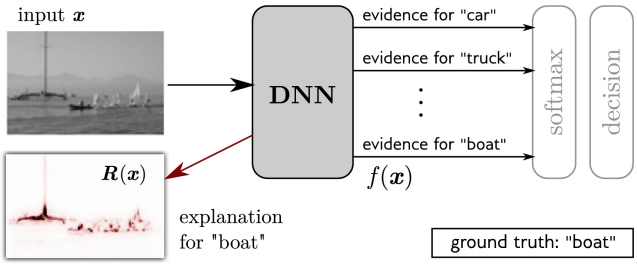

Figure 5: Explanation of the prediction f(x) produced by the DNN for a given data point x. Here, f(x) represents the evidence for the target class (boat) as represented by the corresponding neuron just before the softmax layer

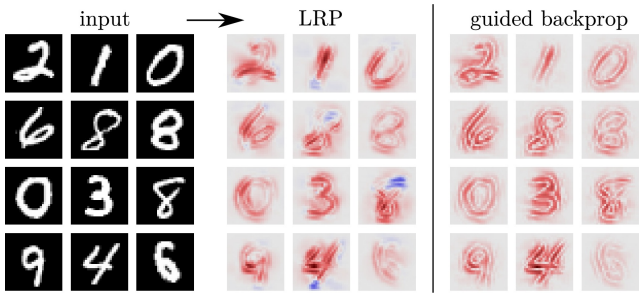

Figure 6: LRP applied to a convolutional DNN trained on MNIST, and resulting explanations for selected digits. Red and blue colors indicate positive and negative relevance scores respectively. Heatmaps are shown next to those produced by guided backprop. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.

4.1 Layer-wise relevance propagation (LRP)

An acceptable approach to explaining the prediction of a DNN is to make explicit use of its graph structure, and proceed as follows: We start at the output of the network. Then, we move in the graph in reverse direction, progressively mapping the prediction onto the lower layers. The procedure stops once the input of the network is reached. Layer-wise mappings can be engineered for specific properties.

Layer-wise relevance propagation (LRP) [2], is applicable to general network structures including DNNs and kernels. The layer-wise mappings are designed to ensure a relevance conservation property, where the share of $f(x)$ received by each neuron is redistributed in same amount on its predecessors. The injection of negative relevance is controlled by hyperparameters.

Example of explanations produced by LRP are given in Fig. 6. Most of the digit contours are identified as relevant, and a few pixels such as the broken upper-loop of the last digit 8 are identified as negatively relevant. In comparison, non-conserving methods such as guided backprop cannot identify these negatively relevant areas.

In the first phase, a standard forward pass is applied to the network and the activations at each layer are collected. In the second phase, the score obtained at the output of the network is propagated backwards in the network. In this layered graph structure, the relevance conservation property can be formulated as follows: Let j and k be indices for neurons of two successive layers. Let R_k be the relevance of neuron k for the prediction $f(x)$. We define $R_{j\leftarrow k}$ as the share of R_k that is redistributed to neuron j in the lower layer. The conservation property for this neuron imposes

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

Likewise, neurons in the lower layer aggregate all relevance coming from the neurons from the higher layer:

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

These two equations, when combined, also ensure a relevance conservation property between layers (proof: $\sum_j R_j = \sum_j \sum_k R_{j \leftarrow k} = \sum_k \sum_k R_{j \leftarrow k} = \sum_k R_k$). Conservation of relevance in the redistribution process also holds globally, so that we can write the chain of equalities

$$\sum_{i=1}^{d} R_i = \cdots = \sum_j R_j = \sum_k R_k = \cdots = f(x)$$

where the leftmost and rightmost terms highlight the fact that the method computes a decomposition of $f(x)$ in terms of input variables.

Technically, this conservation property of LRP must be implemented by a specific set of propagation rules. Let the neurons of the DNN be described by the equation

$$a_k = (\sum_j a_j w_{jk} + b_k)$$

One propagation rule that is locally conservative and that was shown to work well in practice is the -rule [2] given by:

$$R_j = \sum_k \left(\alpha \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} - \beta \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-} \right) R_k \tag{2}$$

where each term of the sum corresponds to a relevance message $R_{j \leftarrow k}$, where $()^+$ and $()^-$ denote the positive and negative parts respectively, and where the parameters and are chosen subject to the constraints $\alpha - \beta = 1$ and $\beta \geq 0$. To avoid divisions by zero, small stabilizing terms can be introduced when necessary. The rule can be rewritten as

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

where $R_k = \alpha R_k$ and $R_k = -\beta R_k$. It can now be interpreted as follows:

Relevance R_k should be redistributed to the lower-layer neurons $(a_j)_j$ in proportion to their excitatory effect on a_k. Counter-relevance R_k should be redistributed to the lower-layer neurons $(a_j)_j$ in proportion to their inhibitory effect on a_k.

Different combinations of parameters α, β were shown to modulate the qualitative behavior of the resulting explanation. As a naming convention, we denote, for example, by LRP-$\alpha_2\beta_1$, the fact of having chosen the parameters $= 2$ and $\beta = 1$ for this rule.

Fig. 7 depicts the redistribution process for a neuron with positive inputs and weights $(w_{jk})_j = (1, 0, -1)$. The higher α and β, the more positive and negative relevance are being created in the propagation phase.

Examples of explanations obtained with different values of and β are given in Fig. 8 for MNIST digits predicted by a convolutional DNN. Unless stated otherwise, we use in all experiments the same parameters and for each hidden layer, except for the first layer.

When $\alpha = 1$, the heatmaps contain only positive relevance, and the latter is spread along the contour of the digits in a fairly uniform manner. When choosing $\alpha = 2$, some regions in the images become negatively relevant (e.g. the broken upper-loop of the last digit 8), but the negative relevance still amounts to only 5% of the total relevance. When setting the higher value $\alpha = 3$, negative relevance starts to appear in a seemingly random fashion, with the share of total relevance surging to 30%. For this simple example, a good choice of propagation rule is LRP-$_2\beta_1$.

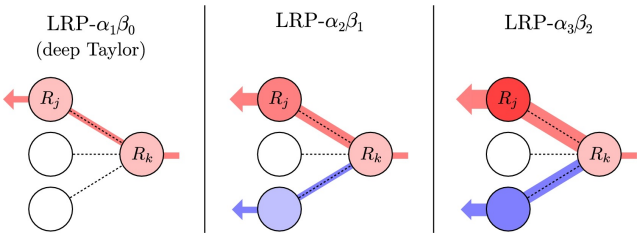

Figure 7: Graphical depiction of the relevance redistribution process for one neuron, with different parameters α and β. Positive relevance is shown in red. Negative relevance is shown in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

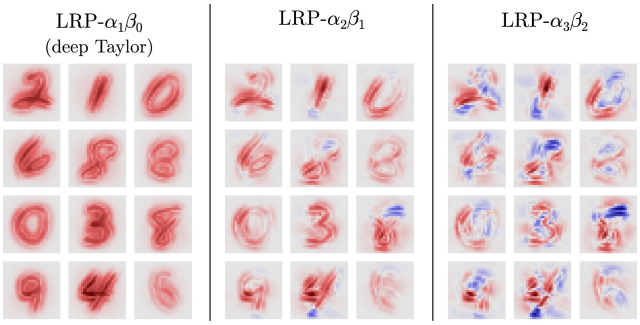

Figure 8: LRP explanations when choosing different LRP parameters α and β. Positive and negative relevance are shown in red and blue respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

On the deeper BVLC CaffeNet for image recognition, LRP-$_2\beta_1$ was also shown to work well [2]. For the very deep GoogleNet, however, LRP-$\alpha_1\beta_0$ was found to be more stable [10].

4.2 Non-conserving backward propagation

Non-conserving backward propagation techniques include deconvolution [18], and its extension guided backprop [20]. Both have been proposed for visualizing the predictions of convolutional DNNs. The first method relies on max-pooling layers to orient the propagated signal on the appropriate locations in the image. The second method relies on the ReLU activations for that purpose. Unlike LRP and other conserving techniques, the visualization produced by these methods cannot directly be interpreted as relevance scores. For comparison purposes, we can however take their absolute values and use them as relevance scores.

Shelhamer et al.2016 [18] show that a fully convolutional network (FCN) trained end-to-end, pixels-to-pixels on semantic segmentation exceeds the state-of-the-art without further machinery. To our knowledge, this is the first work to train FCNs end-to-end (1) for pixelwise prediction and (2) from supervised pre-training. Fully convolutional versions of existing networks predict dense outputs from arbitrary-sized inputs. Both learning and inference are performed whole-image-ata-time by dense feedforward computation and backpropagation. In-network upsampling layers enable pixel-wise prediction and learning in nets with subsampled pooling which is shown in Fig. 9.

4.3 Variational inference

In some real-world prediction cases such like movie rating and web page popularity, a heat-map on the input is not explainable enough. We need the semantic or concise representation of data to support the prediction or decision. This work could be done by Bayesian reference.

One of the core problems of modern statistics is to approximate difficult-to-compute probability densities. This problem is especially important in Bayesian statistics, which frames all inference about unknown quantities as a calculation involving the posterior density. Variational inference (VI),

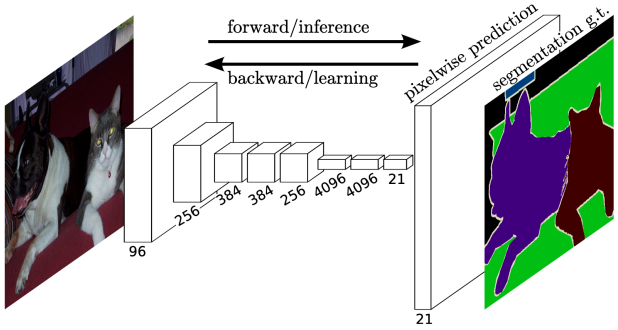

Figure 9: Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

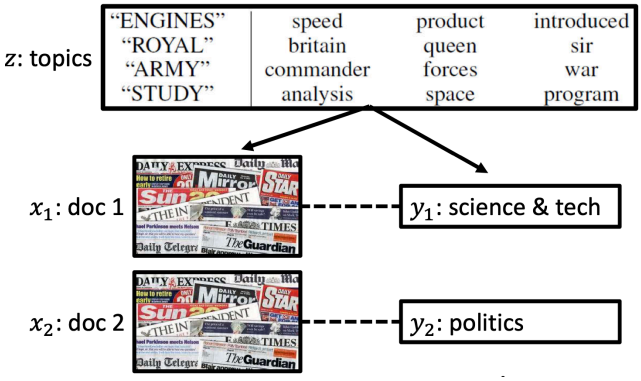

Figure 10: Using supervised LDA to draw semantic or concise representation of data x (via latent variable z)

a method from machine learning that approximates probability densities through optimization.VI has been used in many applications and tends to be faster than classical methods, such as Markov chain Monte Carlo sampling. The idea behind VI is to first posit a family of densities and then to find the member of that family which is close to the target. Closeness is measured by Kullback-Leibler divergence [5],[4], [3].

$$q^*(z) = \underset{q(z)\in Q}{\operatorname{argmin}} \mathrm{KL}(q(z)||p(z|x)) \tag{3}$$

Blei et al. 2010 [4] introduce supervised latent Dirichlet allocation (sLDA), a statistical model of labelled documents. The model accommodates a variety of response types. We derive a maximum-likelihood procedure for parameter estimation, which relies on variational approximations to handle intractable posterior expectations. sLDA was used in drawing semantic or concise representation of document which is shown in Fig. 10 (via latent variable).

We could also image that if a sLDA model is attached on a given DNN (using the output of given DNN as the sLDA's labels), the z shall be the explanation of the decision or prediction.

5 Casual inference

Base on the SCM assumptions [16], using back-door criterion to do the deconfounder work and find the causal effect of the input on the output.

$$P(Y|do(X)) = \sum_z P(Y|X, Z = z)P(Z = z) \tag{4}$$

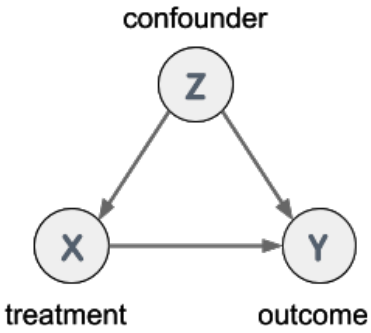

Figure 11: Back-door criteria

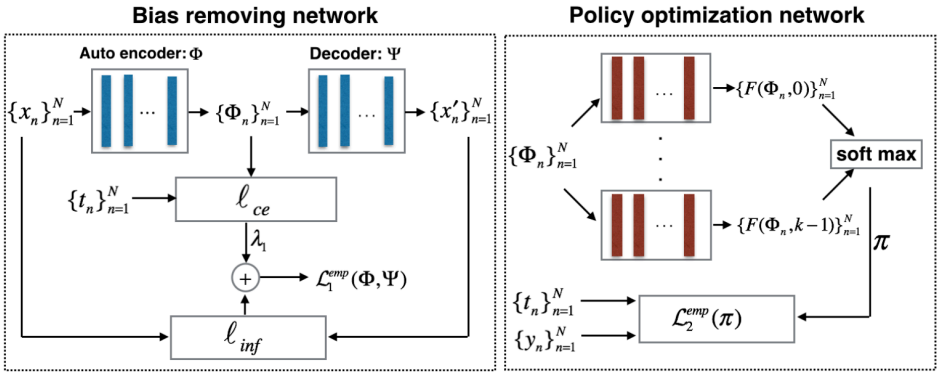

Figure 12: Neural network structure of deconfounder

Atan et al. 2018 [1] propose a novel approach for constructing effective treatment policies when the observed data is biased and lacks counterfactual information. Learning in settings where the observed data does not contain all possible outcomes for all treatments is difficult since the observed data is typically biased due to existing clinical guidelines. This is an important problem in the medical domain as collecting unbiased data is expensive and so learning from the wealth of existing biased data is a worthwhile task. Our approach separates the problem into two stages: first we reduce the bias by learning a representation map using a novel auto-encoder network—this allows us to control the trade-off between the bias-reduction and the information loss—and then we construct effective treatment policies on the transformed data using a novel feedforward network. Separation of the problem into these two stages creates an algorithm that can be adapted to the problem at hand—the bias-reduction step can be performed as a preprocessing step for other algorithms. We compare our algorithm against state-of-art algorithms on two semi-synthetic datasets and demonstrate that our algorithm achieves a significant improvement in performance.

6 Conclusion

Building transparent machine learning systems is a convergent approach to both extracting novel domain knowledge and performing model validation. As machine learning is increasingly used in real-world decision processes, the necessity for transparent machine learning will continue to grow.

This tutorial has covered the following key directions for improving machine learning transparency:

Interpretation

An interpretation is the mapping of an abstract concept (e.g. a predicted class) into a domain that the human can make sense of. Interpretation is find a post probability of a abstract concept of a given DNN (carrying $p(w_c|x)$ information) using Bayesian reference.

Explanation

Explaining the model's decisions and predictions by:

(1) Identifying the interpretive relevant input variables.

(2) Non-conserving backward propagation techniques include deconvolution [18], and its extension guided backprop [20]. Both have been proposed for visualizing the predictions of convolutional DNNs.

(3) Identifying the interpretive relevant latent variables.

Causality

Base on the SCM assumptions, an essential element of correlational study design is to identify potentially confounding influences on the variable under study. The exactly casual effect of input on the output of given DNN could be found out if the confounders are included in the input.

References

[1] O. Atan, J. Jordon, and M. v. d. Schaar. Deep-Treat- Learning Optimal Personalized Treatments from Observational Data using Neural Networks. *Association for the Advancement of Artificial Intelligence*, 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:e0130140, 2015.

[3] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112:859–877, 2016.

[4] D. M. Blei and J. D. McAuliffe. Supervised Topic Models. 2010.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing Higher-Layer Features of a Deep Network. *University of Montreal*, 2009.

[7] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. pages 609–616, 2009.

[8] J. Li, X. Chen, E. Hovy, and D. Jurafsky. Visualizing and Understanding Neural Models in NLP. 2015.

[9] Z. C. Lipton. The mythos of model interpretability. *Communications of the ACM*, 61:36–43, 2018.

[10] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

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

[12] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3510–3520, 2017.

[13] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. 2016.

[14] A. Nguyen, J. Yosinski, and J. Clune. Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks. 2016.

[15] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. 2016.

[16] J. PEARL. Causal diagrams for empirical research. *Biometrika*, 82:669–688, 1995.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin. Why Should I Trust You?: Explaining the Predictions of Any Classifier. pages 1135–1144, 2016.

[18] E. Shelhamer, J. Long, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. 2016.

[19] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. 2013.

[20] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for Simplicity: The All Convolutional Net. 2014.