Methods for Interpreting and Understanding Deep Neural Networks

Paper | Full Notes

In order to be able to understand DNN we need to first differentiate between interpretation and explanation. **Interpretation** is the mapping of an abstract concept (e.g. a vector space) in a human-understandable domain (e.g. an image/text/sound). **Explanation** refers to the collection of features from the interpretable domain that have contributed for a given input to lead to a certain output (decision).

Interpreting the DNN model means that we need to find the corresponding prototype for each of the defined classes and map it in a human-understandable domain. This can be done through **Activation Maximization (AM)**. There are three types of such algorithms, which are **General AM**, **AM with an expert**, **AM in the code space (decoding function)**.

The **AM algorithm** is a framework that finds an input that will produce the highest score for the concept (class) of interest. In its general form, AM can be executed by maximizing the sum of the output from the DNN and the negative L2 regularization term of the input with gradient ascent.

In the **AM with an expert** algorithm, the L2 regularization term is changed with a data density model *log p(x)* (e.g. Gaussian or convolutional RBM (Restricted Boltzmann Machine)). Due to the difficulty of training a good *expert*, another possibility arises in the face of **AM in the Code Space**. Here a second generative model is trained which tries to map entities from the output space into the input space.

In order to find the features that drive the decisions of the DNN, the following methods can be applied.

Sensitivity analysis tries to answer the question "*What makes x more/less representative of the concept Wc?".* Here the relevance scores *Ri* are equal to the gradient squared. The most relevant *neurons* are thus the ones for which the output is most sensitive.

Simple Taylor Decomposition tries to explain the decision of the model by decomposing the output value as a sum of relevance scores *Ri*. The relevance scores are computed by multiplying the *sensitivity* (gradient) with the *saliency* (input) of the neuron.

Relevance propagation decomposes a DNN by starting at the output and redistributing the prediction score going backwards. The weights and activations are divided in two terms based on the sign of the weights. The sum of positive weights is multiplied by the factor alpha and the negative weights are multiplied by the factor beta. The used LRP is defined by the two constants. Popular choices are LRP-a1b0 and LRP-a2b1. The *Deep Taylor LRP* rules for more traditional layer types can be found in the paper, while the handling of more special layers is not agreed upon in the literature. The redistribution of the score should be **conserved**.

In the paper a handful of tricks & recommendations for the LRP framework has been proposed. Among others, one should limit the usage to model types which are known to provide good results with the method such as CNNs. The usage of DENSE layers should be limited. Different configurations for the alpha and beta parameters have been proposed.

The quality of the provided explanations can be quantified with the following metrics.

Explanation Continuity means that similar input data points should have similar explanations. Both sensitivity analysis and the simple Taylor decomposition are discontinuous on the x1 = x2 line, while the deep Taylor LRP is not.

Explanation Selectivity means that we iteratively remove the features of the input with the highest relevance score. The *certainty* of the classification should then, as a result of that, drop sharply. Deep Taylor LRP performs the best out of the three methods

Q's:

- Is the *expert* in the AM algorithm pretrained?

- Can we not use a GAN-like approach in the case of AM in Code Space? If yes, does it not make sense to train the original model in such a fashion so that we have interpretation embedded in the original training process?
- In the right bottom of page 4 how do we derive the equation x^{~} = lim_{epsilon->0}epsilon*x?
- Why the author introduce ck in relevance neuron(Rk=ak*ck)?
- Why can we add a bias term *bk* in our DNN and this even improves the result? In the analysis of Simple Taylor Decomposition, only deep ReLU networks (without biases) fulfill the equation that relevance scores equal the first-order terms.

Feature Visualization

Paper | Full Notes

Feature visualization by optimization works by starting with a random input and slowly tweaking the input until a certain structure of the DNN reaches its maximal possible state. Based on the objective at hand, we can optimize for a neuron, channel, layer, class logits or class probability.
Well-trained DNNs are often diverse in the sense that multiple relatively different instances of a certain class would elicit the same response. In order to address the intra-class diversity, we can add an extra *diversity* term to our objective that pushes examples to differ from one another (e.g. penalize cosine similarity).
Due to the fact that we often end up with strange mixtures of objects when we optimize for a single neuron, a need arises to optimize for a combination of neurons, rather than a single one. In order to do so, we can define the activation space to be all possible combinations of neuron activations. A single activation would then be a basis vector in this space. A combination of neurons would just be a vector residing in the activation space. By using this as a framework, we can generate new images which represent a combination of neurons rather than a single one.
If we want to visualize features by just optimizing an image to make neurons react without any constraints, we often end up with a noisy image full of nonsensical high-frequency patterns, which could be explained by the CONV & POOL nature of the majority of DNNs used in CV. In order to combat this problem, some form of **regularization** should always be provided. The forms of regularization can be categorized in three separate families.
Frequency penalization is targeted at removing the high frequency noise from images. It could either penalize variance between neighboring pixels or blur the image after each optimization step. **Transformation robustness** tries to find only these inputs which would elicit the same response even upon small variations in them. Here, a jitter, rotation or scaling of the image before each optimization step could be used. Instead of applying heuristics we can also use **learned priors**, where we either learn a generator function such as GAN or VAE or learn a prior giving us access to the gradient of probability.
Preconditioning in optimization refers to reducing high frequencies in the gradient. It does not change the minimums, but rather guides the optimization process to certain minimums which might be more favorable. Using different regularization metrics will thus lead to different results. The Linf metric increases the high frequencies, while the decorrelated space decreases them. Furthermore, the decorrelated space often leads to images of better quality and makes the optimization process faster.

Q's

- Is the optimization with diversity examples a result of multiple runs of the same algorithm with a different starting point?
- In the third section, we introduce an activation space in order to provide a framework for combining multiple neuron activations. Consequently, the argumentation comes that the basis vectors (single neuron activations) be more interpretable than random directions (combination of neuron activations). If that is so, why do we need to consider the activation space in the first space?
- What is the decorrelated space referring to?

The Building Blocks of Interpretability

Paper | Full Notes

In order to make sense of the abstract mathematical space of the hidden layers in DNNs, we build **semantic dictionaries**. To do so, we map neuron activations to a visualization of the

neuron and sort them by magnitude. Such dictionaries build the base for the interpretability of DNN.
Building up on the semantic dictionaries, we can combine multiple neurons and their visualizations to reason about the interpretation of the input from the DNN in a region of the input. By doing this iteratively, we deduce that deeper layers can find more sophisticated structures in the input.
While the visualization shows what a DNN can detect, they still do not provide any information about the way DNNs take decisions. The most common interface for attribution is called a **saliency map**, which highlights the regions of the input which contribute the most to the final decision of the DNN. This method exhibits two problems. First, pixels are often entangled with other pixels and provide little connection to the high-level concepts. Secondly, they don't allow for probing into the hidden layers. In order to improve on that, we can apply the method not only to the input, but also to the hidden layers to understand the gradual conceptualizing of classes that the DNN undertakes.
An alternate method is the **channel attribution**, by which we slice the cube by channels rather than spatial positions, so that we can find out the role of each filter/channel/detector in the final decision.
One of the main issues that arises by trying to provide a human-understandable interpretation of DNNs is the sheer scale of modern DNNs, which normally consist of millions of neurons spread across tens or even hundreds of layers, each of which consisting of hundreds of channels. Trying to implement previously discussed techniques leads to enormous amount of information that an user should be able to follow in order to understand a single decision of a DNN. In order to prevent that, we can group neurons together by using matrix factorization in order to make the understanding of the higher-level concepts that the DNN has learned easier for a human. These groups can spawn in both spatial, but also channel direction. These groups will then be the atomar structures on the base of which interfaces for the interpretation of DNNs could be constructed.
An interface is an union of elements, which display a specific type of content (e.g. activations or attribution) using a corresponding style of presentation (e.g. feature visualization). The content of elements lives in substrates defined by the decomposition of layers into atoms (e.g. neurons, channels, groups, etc). Each interface should be constructed in such a way that it answers what the DNN recognizes, how it develops understanding or focus on making the things human-scale.
Furthermore, the provided set of building blocks is not exhaustive and can be further extended by other dimensions such as the role of the dataset or the model parameters. Such interfaces could also be used for comparing models or seeing the evolution of the decision-making process during the learning phase of the DNN.
Based on earlier research, it can be concluded that directions in DNN are semantically meaningful. Furthermore, neurons have been found to respond to a mixture of concepts, rather than a single one. A concern in the case of attribution is the **"path-dependance",** which, however, turns out to not have a significant influence on the output.

Q's

- How are spatial and channel attribution different? Does the spatial one only concerns itself with a single channel or positions across all channels?

Tutorial on Interpretable Machine Learning

Paper | Video | Full Notes

Overview
Different techniques: **Sensitivity**, **Deconvolution**, **LRP**, and **Friends**
Perspectives to understand Deep Nets:

- Mechanistic Understanding: Understand how each neuron works
- Functional Understanding: Understand how the network maps the input to output

- Model Analysis: In a view of whole ensemble of the data (average)
- Decision Analysis: In a single decision view (individual)

Model Analysis

1. Class Prototypes
   How does the appropriate class prototype look like for the whole ensemble.
   argmaxf(x) to find the typical look according to the neural network.

Decision Analysis

1. Sensitivity Analysis
   Analyze neural network by computing the partial derivative of the overall function w.r.t xi (gradient of every **pixel**).
   Q: Which pixels lead to increase/decrease of prediction score when **changed**?
   Problems:
   - Sensitivity analysis doesn't highlight pixels corresponding to cars -> Explain the variation of the function not the function itself (e.g. Highlight pixels make the image more or **less** a car)
   - Shattered Gradients: Input gradient becomes increasingly highly varying and unreliable with neural network depth
2. Individual Explanations (LRP)
   Why is a **given image** classified as a car? *heatmap = LRP(x,f)*
   Q: Which pixels **contribute how much** to the classification?
   LRP resolve the shattered gradients issue from not computing **gradient x inputs**

LRP

Steps

1. Initialize relevance of last layers *rj* to results of the classification *rj = f(x)*
2. Compute the relevance from upper layers using simple LRP rule
3. Relevance Conservation Property: Sum of relevance in every layers is constant (*f(x)*).

Axiomatic Approach to interpretability: Ground truth (Highlighting pixels that deep neural network comes up for car) is hard to obtain for multiple classes -> Evaluate the explanation technique **axiomatically** (e.g. It must pass a number of predefined **unit tests**)

Properties of Heatmap

1. Conservation: Total attribution on the input features should be proportional to the amount of (explainable) evidence at the output. Use LRP-a1b0 propagation rule to prove.
2. Positivity: If the neural network is certain about its prediction, input features are either relevant (positive) or irrelevant (zero)
3. Continuity: If two inputs are almost the same(a image of car and a image of car with slight perturbation), and the prediction is almost the same, then the explanation should also be almost the same. On the other hand, **Gradient based methods** (e.g. Sensitivity Analysis) produce discontinuous explanation, because their heatmaps tend to flicker at much higher frequency than images are actually changing.
4. Selectivity: If input features are attributed relevance, removing them should reduce evidence at the output (Test selectivity with Pixel-Flipping)

LRP-a1b0 has all these four properties

From LRP to Deep Taylor Decomposition

The LRP-a1b0 can be seen as a deep Taylor decomposition (DTD) which yields domain- and layer- specific propagation rules.

Proposition: Relevance at **each layer** which is propogated from the top layer to the input layer is a product of the activation and an approximately constant term.

1. Build the Relevance Neuron: Relevance *Rj* can be seen as a function of the previous layer. *cj* modulation term which is constant and positive.
2. Expand the Relevance Neuron: Analyze the previous max function (relu) using Taylor decomposition (near zero of the relu function), which will give us what is the proper weight to redistribute from one neuron to the neuron of the previous layer. Relevance

as a fucntion of the activation in the previous layer = relevance evaluated in the root point + a sum of the first oder Taylor expansion terms + epsilon.

3. Decompose Relevance: *Rj(root point)=0* because the root point is near zero. The middle term is the **sum of all input features** -> Identify what are the importance of each input neurons for producting relevance in the higher layer
   How to choose the root point
   1. nearest root: Descend along the direction of gradient of the relevant neuron until hit the root point
   2. rescaled activation: Decend along the direction of activation which is rescaled towards the origin
   3. **rescaled excitations**: Following 2. + Descend in the input domain only along directions that have positive weights -> Gives the generic LRP-a1b0 rule. Root point belongs to the input domain of activation function + root point is close to the actual activation pattern.
4. Application to input layers:
   1. Choose a root point that is nearby and satisfies domain constraints
   2. Inject it in the generic DTD rule to get the specific rule
5. Pooling relevance over all outgoing neurons: Treat pooling layers as relu detection layers

General pipeline for CNN: **Add relu at the last layer** + **use LRP-a1b0 whether its pulling or convolution** + **DTD for pixels at the first layer**

Q's

- I suggest we have a discussion about the Deep Taylor Decomposition as this is the most abstract concept among all the papers.