InterpretDL: Explaining Deep Models in PaddlePaddle

Xuhong Li
lixuhong@baidu.com

Haoyi Xiong∗
xionghaoyi@baidu.com

Xingjian Li
lixingjian@baidu.com

Xuanyu Wu
v_wuxuanyu@baidu.com

Zeyu Chen
chenzeyu01@baidu.com

Dejing Dou
doudejing@baidu.com

Baidu, Inc., Beijing, China

Editor: Alexandre Gramfort

1. Background

Deep neural networks (LeCun et al., 2015) are well-known for their excellent performance in handling various machine learning and artificial intelligence tasks. Meanwhile, they are also known for their black-box natures (Castelvecchi, 2016); it is often difficult for humans to understand the prediction results of these deep models. In recent years, many interpretation tools (Ribeiro et al., 2016; Selvaraju et al., 2017; Smilkov et al., 2017; Sundararajan et al., 2017; Toneva et al., 2019; Wang et al., 2020) have been proposed to explain or reveal the ways that deep models make decisions, in order to identify \textit{whether the deep models work} and \textit{why they work}. While most interpretation algorithms for explaining deep models have been already open-sourced, there are still several issues and technical burdens for industrial and academic practices.

First of all, deep models in various domains (e.g., structural data, images, and texts) desire varying types of explanations. For example, one could easily explain the prediction results of deep models with structural data through finding the important features/variables
in the given data point, while the interpretation algorithms for image classification models use discriminative pixels (Smilkov et al., 2017) or superpixels (Ribeiro et al., 2016) for visual explanations. Then, it lacks a unified abstraction layer (e.g., a standard set of Application Programming Interfaces (APIs)) for the existing interpretation algorithms. Thus, there is a need to retrieve, read, and comprehend a long list of documents and even source codes to use an interpretation algorithm. Finally, most interpretation algorithms were implemented independently, in an ad-hoc manner, based on different programming frameworks of deep learning (e.g., Pytorch and Tensorflow). Though some efforts (Alber et al., 2019; tfe; Kokhliyan et al.) have been done to collect and unify a number of explanation algorithms for Tensorflow or Pytorch, still, it would be heavy to explain deep learning models that are trained on other frameworks.

2. The Proposed Toolkit InterpretDL

To cope with these issues, we report InterpretDL, an open-source library with mainstream explanation algorithms for deep models’ explanations based on PaddlePaddle (Ma et al., 2019), which is a deep learning framework with over 4 million developers currently and numerous users. Model explanations are one of the most desirable features for the PaddlePaddle ecosystem. To avoid unexpected and unnecessary switch-framework efforts, we provide the easy-to-use tool InterpretDL for explaining the behaviors of PaddlePaddle deep models. We follow the plug-and-play fashion and use universal and straightforward APIs, with at least one detailed tutorial for each algorithm, towards a user-friendly library for both industrial and academic practices.

The proposed toolkit InterpretDL also considers the compatibility across frameworks, by modularizing the explanation algorithms where the implementations concerning the framework are wrapped into one single module. In this way, any other module that returns desired outputs would be compatible with InterpretDL. We have currently supported this feature for most gradient-based algorithms (Smilkov et al., 2017; Lundberg and Lee, 2017) and model-agnostic ones (Ribeiro et al., 2016; Ahern et al., 2019). We will continuously support this compatibility across frameworks for more explanation algorithms.

Furthermore, we provide a new categorization for explanation algorithms. As shown in Figure 1, we show a tree-based taxonomy for the 18 current (and planned) algorithms
InterpretDL: Explaining Deep Models in PaddlePaddle

from interpretdl import SmoothGradInterpreter

from paddle.vision.models import resnet50
paddle_model = resnet50(pretrained=True)

# Or load a custom model.
# paddle_model = a_custom_model()
# model_path = "path/to/trained_weights.pdparams"
# paddle_model.set_dict(paddle.load(model_path))

sg = SmoothGradInterpreter(paddle_model, device="gpu:0")
gradients = sg.interpret("test.jpg", visual=True, save_path=None)

Listing 1: Codes for SmoothGrad-based explanations of ResNet-50 based on a given image.

in InterpretDL with the properties of algorithms, i.e., “proxy”, “derivation” and “closed-form”, following Li et al. (2021). The algorithms were chosen according to their popularity, efficiency, effectiveness and potentials. Compared to existing deep learning interpretability libraries (tfe; Kokhliyan et al.; Alber et al., 2019), InterpretDL considers a wider concept of explanations. Training-phase explanations are included for investigating the insights to understand the training process and the influences or contributions of the training set to the trained model. As for post-hoc explanations, we provide both global and local explanation algorithms. More specifically, we provide three different sub-categories of local ones beyond using vanilla gradients and activations.

The trustworthiness is also one important property of explanation algorithms that assures the explanation results be faithful to the model. InterpretDL currently provides three methods to validate the trustworthiness, based on perturbation tests (Samek et al., 2016; Petsiuk et al., 2018) and localization ability (Zhang et al., 2018).

Comparisons to Existing Interpretation Libraries  Compared to the existing open-sourced libraries of interpretation/explanation algorithms for machine learning in general, e.g., InterpretML (Nori et al., 2019), AIX360 (Arya et al. 2020) and Alibi (Klaise et al., 2021), InterpretDL is designed and optimized for interpreting deep neural networks, which would be computational intensive with the design and implementation of specific deep models. For interpreting deep models, Captum (Kokhliyan et al.) provided post-hoc local explanation algorithms based on Pytorch (Paszke et al., 2019), tfe-explain (tfe) for Tensorflow (Abadi et al., 2016) models, and similarly iNNvestigate (Alber et al., 2019) for Keras with TensorFlow backend. Compared to them, the proposed toolkit InterpretDL considers a wider concept for explanations, containing both training-phase explanations (Toneva et al., 2019) and post-hoc ones (Ahern et al., 2019; Wang et al., 2020), and focuses on the PaddlePaddle ecosystem. Meanwhile, InterpretDL supports cross-framework compatibility, through modularized designs and simple, clear, and unified APIs.

3. InterpretDL Examples with Codes and Results

To demonstrate the efficiency and effectiveness of using InterpretDL to interpret the prediction results of deep models, we provide two examples, including code and visualization results. Listing 1 shows lines of codes for interpreting a trained ResNet-50 model, where the classification result of the model based on the image stored in “test.jpg” is explained using SmoothGrad algorithm (Smilkov et al., 2017). Figure 2 (b) shows the SmoothGrad
Figure 2: Explanations of a ResNet-50 model based on a given image using three algorithms: SmoothGrad (b), where the explanation scores are added to the “green” channel; LIME (c), where a threshold of top 20% superpixels is applied; GradCAM (d), where the low-resolution explanation is resized to the original image scale and the “JET” color map is applied.

Figure 3: Interpretation results of a sentiment classifier based on texts.

InterpretDL is hosted on GitHub under Apache License 2.0. The package can be easily installed from Python Package Index (PyPI) or from the source repository for development/developer usages. InterpretDL provides the documentation for each implemented function, which is compatible to the sphinx documentation generator, enabling the automatic documentation generation. Interactions with the community, including pull requests and proposals of issues, can be done within the GitHub website with regular updates. InterpretDL follows the PEP8 style for Python codes. Unit tests with a line coverage of 93% of the code and the continuous integration guarantee the code quality. In the future, we would continue implementing and integrating new interpretation algorithms into InterpretDL, with the same APIs and designs. An interface is also planned to facilitate visual comparisons. Other valuable components are also in consideration.

1. Available online: https://github.com/PaddlePaddle/InterpretDL
Acknowledgments

The first two authors are supported in part by National Key R&D Programs of China under the grant No. 2021ZD0110303.

References


