CAPTUM: A UNIFIED AND GENERIC MODEL INTERPRETABILITY LIBRARY FOR PYTORCH

Narine Kokhlikyan[*] **Vivek Miglani**[*] **Miguel Martin** **Edward Wang** **Bilal Alsallakh**
Facebook AI
{narine,vivekm,miguelmartin,hack,bilalsal}@fb.com

Jonathan Reynolds **Alexander Melnikov**[†] **Natalia Kliushkina** **Carlos Araya**
Facebook AI
{jonreynolds,sanekmelnikov,natalial,caraya}@fb.com

Siqi Yan **Orion Reblitz-Richardson**
Facebook AI
{siqi,orionr}@fb.com

ABSTRACT

In this paper we introduce a novel, unified, open-source model interpretability library for PyTorch (Paszke et al., 2019). The library contains generic implementations of a number of gradient and perturbation-based attribution algorithms, also known as feature, neuron and layer importance algorithms, as well as a set of evaluation metrics for these algorithms. It can be used for both classification and non-classification models including graph-structured models built on Neural Networks (NN). In this paper we give a high-level overview of supported attribution algorithms and show how to perform memory-efficient and scalable computations. We emphasize that the three main characteristics of the library are multimodality, extensibility and ease of use. Multimodality supports different modality of inputs such as image, text, audio or video. Extensibility allows adding new algorithms and features. The library is also designed for easy understanding and use, and is a valuable tool for building Responsible AI frameworks.

1 INTRODUCTION

Given the complexity and black-box nature of NN models, there is a strong demand for clear understanding of how those models reason. Model interpretability aims to describe model internals in human understandable terms and is an important component of Explainable (Gilpin et al., 2018) and Responsible (Barredo Arrieta et al., 2020) AI. While building interpretable models is encouraged, many existing state-of-the-art NNs are not designed to be inherently interpretable, thus the development of algorithms that explain black-box models becomes highly desirable. This is particularly important when AI is used in domains such as healthcare, finance or self-driving vehicles where establishing trust in AI-driven systems is critical.

Some of the fundamental approaches that interpret black-box models are feature, neuron and layer importance algorithms, also known as attribution algorithms. Existing frameworks such as DeepExplain (Ancona et al., 2018), Alibi (Klaise et al., 2019) and InterpretML (Nori et al., 2019) have been developed to unify those algorithms in one framework and make them accessible to all machine learning model developers and practitioners. These frameworks, however, have insufficient support for PyTorch models with different modalities. In Captum, we provide generic implementations of a number of gradient and perturbation-based attribution algorithms that can be applied to any PyTorch

[*]Equal Contribution
[†]Work done while at Facebook AI

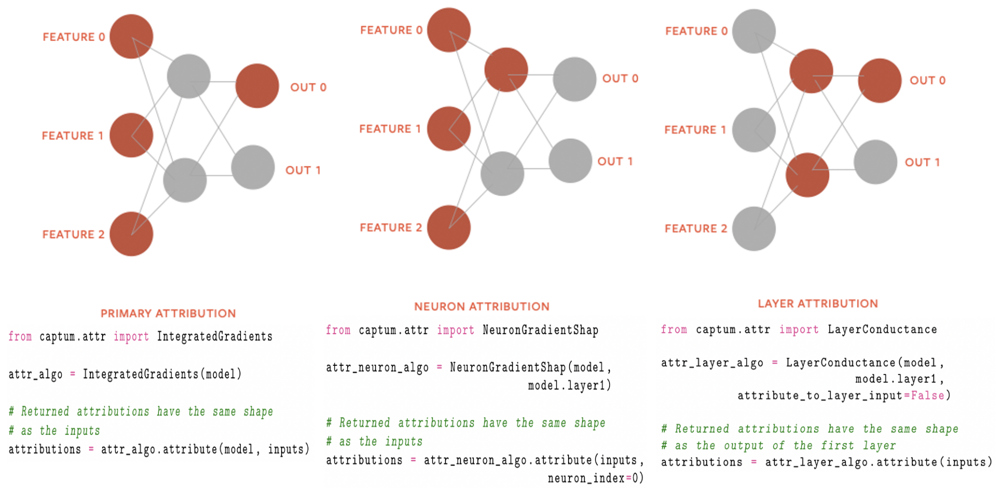
model of any modality. The library is easily extensible and lets users scale computations across multiple GPUs and handles large-sized input by dividing them into smaller pieces, thereby preventing out of memory situations.

Another important aspect is that both qualitative and quantitative evaluation of attributions are difficult. Visual explanations can be misleading (Adebayo et al., 2018) and evaluation metrics are subjective or domain specific (Yang & Kim, 2019). To address these issues, we provide generic implementations of two evaluation metrics called infidelity and max-sensitivity proposed in (Yeh et al., 2019). These metrics can be used in combination with any PyTorch model and most attribution algorithms.

Lastly, model understanding research largely focuses on the Computer Vision (CV) domain whereas there are many unexplored NN applications that desperately need model understanding tools. Adapting CV-specific implementations for those applications is not always straightforward, thus the need for a well-tested and generic library that can be easily applied to multiple domains across research and production.

2 An Overview of the Algorithms

The attribution algorithms in Captum can be grouped into three main categories: primary-, neuron- and layer- attributions, as shown in Figure 1. Primary attribution algorithms are the traditional feature importance algorithms that allow us to attribute output predictions to model inputs. Neuron attribution methods allow us to attribute an internal, hidden neuron to the inputs of the model and layer attribution variants allow us to attribute output predictions to all neurons in a hidden layer. In most cases, both neuron and layer variants are slight modifications of the primary attribution algorithms.



```
from captum.attr import IntegratedGradients

attr_algo = IntegratedGradients(model)

# Returned attributions have the same shape
# as the inputs
attributions = attr_algo.attribute(model, inputs)
```

```
from captum.attr import NeuronGradientShap

attr_neuron_algo = NeuronGradientShap(model,
                                      model.layer1)

# Returned attributions have the same shape
# as the inputs
attributions = attr_neuron_algo.attribute(inputs,
                                          neuron_index=0)
```

```
from captum.attr import LayerConductance

attr_layer_algo = LayerConductance(model,
                                   model.layer1,
                    attribute_to_layer_input=False)

# Returned attributions have the same shape
# as the output of the first layer
attributions = attr_layer_algo.attribute(inputs)
```

Figure 1: An overview of all three types of attribution variants with example code snippets. The first variant depicted on the far-left side of the diagram represents primary attribution. The middle one represents neuron and the right-most one layer attribution.

Most attribution algorithms in Captum can be categorized into gradient and perturbation-based approaches as also depicted in Figure 2. Some of these algorithms such as GradCam (Selvaraju et al., 2017) and Occlusion (Zeiler & Fergus, 2014) are more popular in the CV community, where they stem from. However, our implementations of those algorithms are generic and can be applied to any model that meets certain requirements dictated by those approaches. For example, GradCam and GuidedGradCam are applicable only to convolutional models.

NoiseTunnel includes generic implementations of SmoothGrad, SmoothGrad Square and VarGrad smoothing techniques proposed in (Smilkov et al., 2017). These methods help to mitigate noise in the attributions and can be used in combination with most attribution algorithms depicted in Figure 2.

The attribution quality of a number of algorithms such as Integrated Gradients (IG) (Sundararajan et al., 2017), DeepLift (Shrikumar et al., 2017), SHAP variants (Lundberg & Lee, 2017), Feature Ablation, LIME (Ribeiro et al., 2016) and Occlusion (Zeiler & Fergus, 2014) depend on the choice of baseline, also known as reference, that needs to be carefully chosen by the user. Baselines express the absence of some input feature and are an integral part of many feature importance equations. For example, black and white images or the average of those two are common baselines for image classification tasks.

From the implementation and usage perspective, all algorithms follow a unified API and signature. This makes it easy to compare the algorithms and switch from one attribution approach to another. The code snippets in Figure 1 demonstrate examples of how to use primary, neuron, and layer attribution algorithms in Captum.

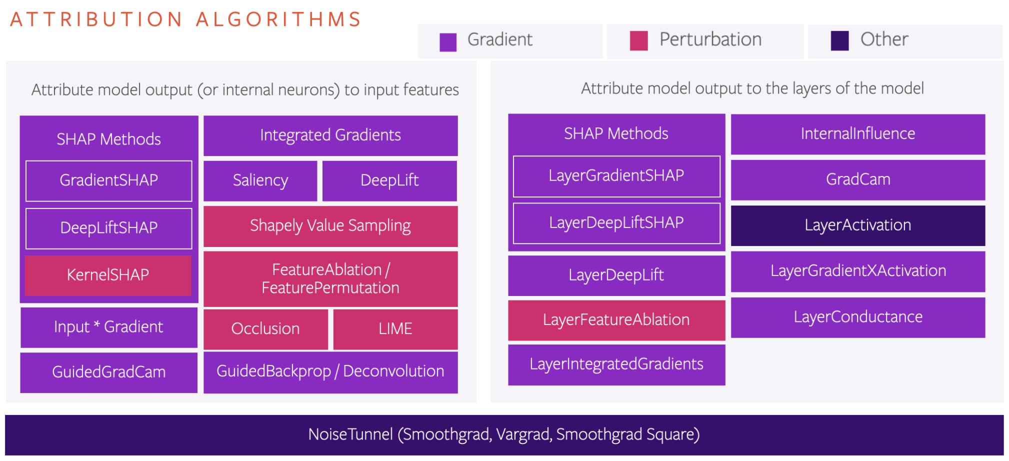


Figure 2: An overview of all the attribution algorithms in Captum. The algorithms grouped on the left side of the diagram are the primary and neuron attribution algorithms. The ones on the right side of the diagram are layer attribution variants. Besides that we can also recognize color-coding of purple for gradient, pink for perturbation and dark blue for algorithms that are neither perturbation nor gradient-based.

2.1 SCALABILITY

Many state-of-the-art Neural Networks with a large number of model parameters use large-sized inputs which, ultimately, lead to computationally expensive forward and backward passes. We want to make sure that we leverage available memory and CPU/GPU resources efficiently when performing attribution. In order to avoid out of memory situations for certain algorithms, especially the ones with internal input expansion, we slice inputs into smaller chunks, perform the computations sequentially on each chunk and aggregate resulting attributions. This is especially useful for algorithms such as IG (Sundararajan et al., 2017) and Layer Conductance (Dhamdhere et al., 2019) because they internally expand the inputs based on the number of integral approximation steps. Being able to chunk the inputs into smaller sizes can theoretically allow us to perform integral approximation for an infinite number of steps.

In case of feature perturbation, if the inputs are small and we have enough memory resources available, we can perturb multiple features together in one input batch. This requires that we expand the inputs by the number of features that we perturb together and helps to improve runtime performance of all our feature perturbation algorithms.

In addition to this, all algorithms support PyTorch DataParallel, which performs model forward and backward passes simultaneously on multiple GPUs and improves attribution runtime significantly. We made this available for all layer and neuron attribution algorithms, including Layer Activation and NoiseTunnel.

In one of our experiments, we used IG on a pre-trained VGG19 model, for a single 3 x 224 x 224 input image in a GPU environment that has 8 GPUs, each 16GB memory available and gradually

Table 1: Runtime performance of IG and Feature ablation for image classification with VGG19 and IG for segmentation with ResNet 101 on a single input image.

Number of GPUs	**Classification with VGG19**		**Segmentation with ResNet 101**
	Int. Grads	Feature Abl.	Int. Grads
1	41.62	140.4	90.49
4	26.41	38.39	32.14
8	15.06	22.02	21.45

increased the number of GPUs while keeping the number of integral approximation steps constant (in this case 2990). From the second column of Table 1 we can see how the execution time decreases substantially as we increase the number of GPUs. In the second experiment we used the same execution environment, pre-trained model, input image and performed feature ablation by ablating multiple features in one batch using a single forward pass. Based on the experimental results shown in the third column of Table 1 we can tell that by increasing the number of GPUs from 1 to 8 the execution time drops by approximately 85%. In the third experiment we used pre-trained ResNet 101 backbone segmentation model and computed IG on segment prediction for a single 3 x 640 x 958 input image. Similar to classification use case, in segmentation example we observe significant runtime performance reduction when we gradually increase the number of GPUs.

3 Evaluation

As mentioned in the sections above, both qualitative and quantitative evaluation of feature, neuron, and layer importances are challenging research areas. Human annotations based on visual perception are often subjective, and many quantitative evaluation metrics rely on priors of a dataset domain. We implemented two metrics, infidelity and maximum sensitivity proposed in Yeh et al. (2019), in a generic manner for use with any PyTorch model and most Captum algorithms. Similar to other algorithms, we can perform multiple perturbations simultaneously which helps to improve runtime performance.

Infidelity is the generalization of sensitivity-n (Ancona et al., 2018) metric that relies on the completeness axiom, that states that the sum of the attributions is equal to the differences of the NN function at its input and baseline. The other metric, called maximum sensitivity, measures the extent of the attribution change when the input is slightly perturbed. Maximum sensitivity is measured using Monte-Carlo approximation by sampling multiple samples from an L_p ball using a perturbation radius $r \in \mathbb{R}$. The users of Captum can use any perturbation function that computes perturbations within any L_p ball. We provide a default implementation that samples uniformly from L_∞ ball. In the appendix we demonstrate results of these two metrics in different applications.

4 Conclusion

We presented a unified model interpretability library for PyTorch, called Captum, that supports generic implementations of a number of gradient and perturbation-based attribution algorithms. Captum can be applied to NN models of any type and used both in research and production environments. Furthermore, we described how the computations are scaled and how we handle large-sized inputs. In addition to that, we also added support for two generic quantitative evaluation metrics called infidelity and maximum-sensitivity.

5 Future Work

Our future work involves both expanding the list of attribution algorithms and looking beyond attribution methods for model understanding. Beyond feature, neuron and layer attribution we are also looking into adversarial robustness and the intersection between these two fields of research. Concept-based model interpretability that aims to explain the models globally using human understandable concepts is another interesting direction to explore.

References

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 9505–9515. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/8160-sanity-checks-for-saliency-maps.pdf`.

Marco B Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *ICLR*, 2018.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.012. URL `https://www.sciencedirect.com/science/article/pii/S1566253519308103`.

Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. How important is a neuron? In *ICML*, 2019. URL `https://openreview.net/pdf?id=SylKoo0cKm`.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 80–89, 2018.

David Harrison and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81 – 102, 1978. ISSN 0095-0696. doi: https://doi.org/10.1016/0095-0696(78)90006-2. URL `http://www.sciencedirect.com/science/article/pii/0095069678900062`.

Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi: Algorithms for monitoring and explaining machine learning models, 2019. URL `https://github.com/SeldonIO/alibi`.

Scott M Lundberg and Su-In 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 (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, abs/1906.00091, 2019. URL `https://arxiv.org/abs/1906.00091`.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*, pp. 1135–1144, 2016.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3145–3153. JMLR.org, 2017.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *ArXiv*, abs/1706.03825, 2017.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3319–3328. JMLR.org, 2017.

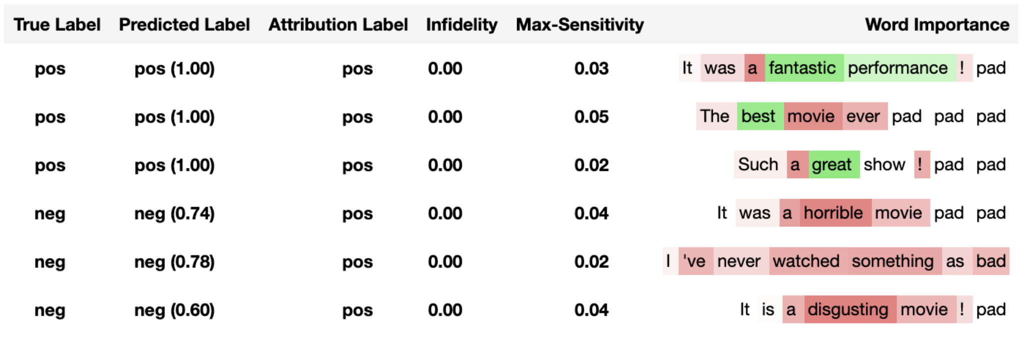
Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance, 2019.

Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David I. Inouye, and Pradeep D. Ravikumar. On the (in)fidelity and sensitivity of explanations. In *NeurIPS*, 2019.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 818–833, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10590-1.

A Applications

The generality of Captum library allows us to apply it to different types of NNs. One of the commonly used applications is text classification. We used a pre-trained text classification model on IMDB dataset (Maas et al., 2011) and IG to identify most salient tokens in the text. Figure 3 visualizes color-coded contributions of each token to the output predictions. We also computed infidelity and max-sensitivity scores. The perturbations for infidelity metric are sampled uniformly from a normal distribution $\mathbb{N}(0, 0.03)$ and for max-sensitivity from a L_∞ ball with a radius of 0.03. The norm $||.||$ used in max-sensitivity computations is the L_2 norm.



True Label	Predicted Label	Attribution Label	Infidelity	Max-Sensitivity	Word Importance
pos	pos (1.00)	pos	0.00	0.03	It was a fantastic performance ! pad
pos	pos (1.00)	pos	0.00	0.05	The best movie ever pad pad pad
pos	pos (1.00)	pos	0.00	0.02	Such a great show ! pad pad
neg	neg (0.74)	pos	0.00	0.04	It was a horrible movie pad pad
neg	neg (0.78)	pos	0.00	0.02	I 've never watched something as bad
neg	neg (0.60)	pos	0.00	0.04	It is a disgusting movie ! pad

Figure 3: Visualizing salient tokens computed by IG that contribute to the predicted class using a binary classification model trained on IMDB dataset. Green means that those tokens pull towards and red that they pull away from the predicted class. The intensity of the color signifies the magnitude of the signal.

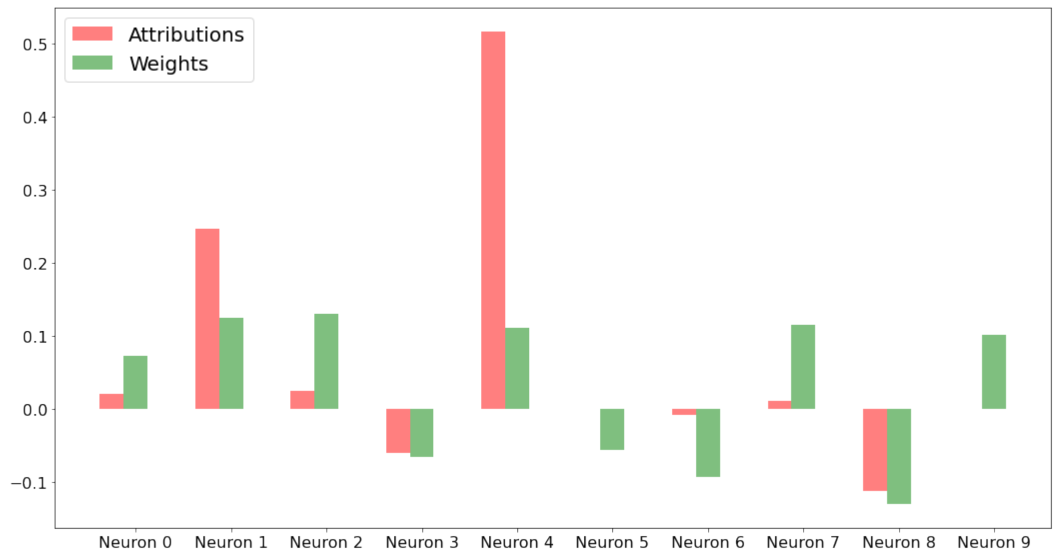
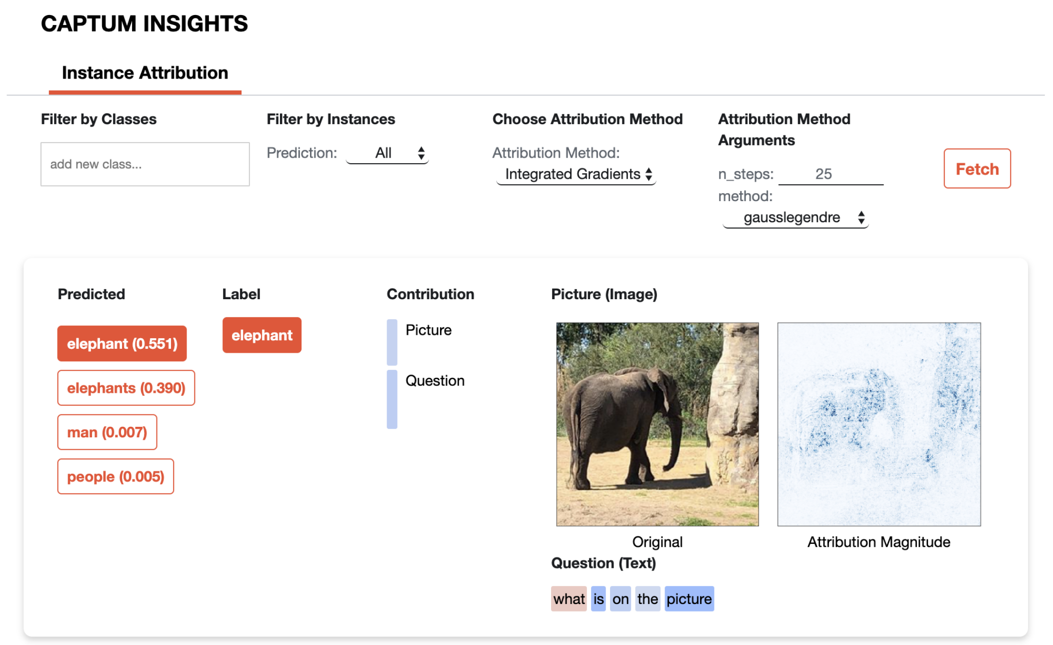


Figure 4: Visualizing normalized attribution scores and weights for all ten neurons in the last linear layer of a simple four MLP model trained on Boston house prices dataset.


CAPTUM INSIGHTS

Instance Attribution

Filter by Classes

add new class...

Filter by Instances

Prediction: All

Choose Attribution Method

Attribution Method: Integrated Gradients

Attribution Method Arguments

n_steps: 25

method: gausslegendre

Fetch

Predicted	Label	Contribution
elephant (0.551)	elephant	Picture
elephants (0.390)		Question
man (0.007)		
people (0.005)		

Picture (Image)

Original

Attribution Magnitude

Question (Text)

what is on the picture

Figure 5: Captum Insights interactive visualization tool after applying IG to the Visual Question Answering multi-modal model. The tool also visualizes aggregated attribution magnitudes of each modality.

Classification models are not the only types of models that Captum supports. As an example, we built a regression model using Boston house prices dataset (Harrison & Rubinfeld, 1978) and a simple four layer NN using linear layers and ReLUs. We attributed the output predictions to the last linear layer using layer conductance algorithm and plotted them together with the learned weights

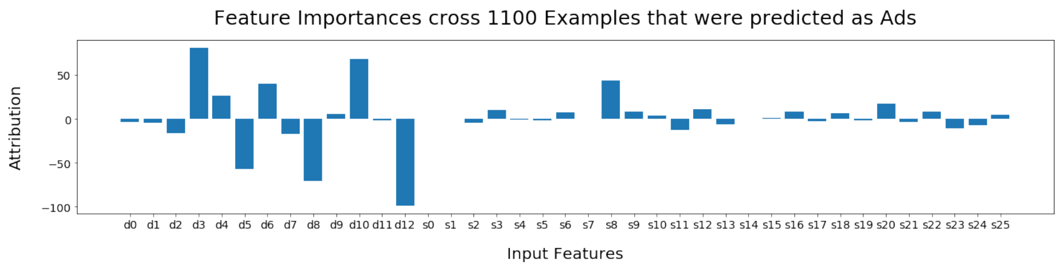


Figure 6: Feature importance scores for dense(d0 - d12) and sparse(s0 - s25) features for DLRM Ads click prediction model.

of that same layer as shown in Figure 4. Here we can see that the weights and the attribution scores are aligned with each other. Both scores were normalized using L_1 norm.

In order to improve model debugging experience, we developed an interactive visualization tool called Captum Insights. The tool allows to sub-sample input examples and interactively attribute different output classes to the inputs of the model using different types of attribution algorithms depicted in Figure 6.

A.1 MULTIMODALITY

The support for multi-modal Neural Networks is one of the core motivations of Captum library. This allows us to apply Captum to machine-learning models that are built using features stemming from different sources such as audio, video, image, text, categorical or dense features. Aggregated feature importance scores for each input modality can reveal which modalities are most impactful. In order to make those visualizations interactive we built a visualization tool on top of Captum called Captum Insights. Captum Insights visualizes feature importance metrics for a selected feature importance algorithm, dataset and a predicted or ground truth label.

In Figure 6 we can see a screenshot of Captum insights tool for multi-modal Visual Question Answering model. From those visualizations we can tell whether the stronger predictive signal is coming from text or image.

Deep Learning Recommendation Models (DLRMs) are another example of multi-model models that are built on dense and sparse (categorical) features. We used a DLRM model trained on Criteo's[1] traffic dataset and a model architecture described in Naumov et al. (2019). Figure 5 visualizes both dense and sparse feature importance calculated with IG and aggregated across 1100 Ad examples. In this example we can see whether stronger prediction signal is coming from dense or sparse features and how is that signal correlated with the Ads Clicked prediction.

[1]kaggle.com/c/criteo-display-ad-challenge