

TorchLens: A Python package for extracting and visualizing hidden activations of PyTorch models

JohnMark Taylor (jt3295@columbia.edu)

Zuckerman Institute, 3227 Broadway
New York, NY 10027 USA

Nikolaus Kriegeskorte (nk2765@columbia.edu)

Zuckerman Institute, 3227 Broadway
New York, NY 10027 USA

Abstract:

Extracting hidden layer activations from deep neural networks (DNNs) is an essential step in characterizing how they transform their internal representations over processing, and in evaluating them as models of brain computation. Here we introduce *TorchLens*, an open-source Python package for extracting hidden layer activations from DNNs implemented in PyTorch. Uniquely among existing approaches to this problem, *TorchLens* has the following features: (1) it exhaustively extracts the outputs and accompanying metadata of all intermediate operations, not just those associated with PyTorch module objects, yielding a full record of every step in the model's forward pass, (2) it can automatically generate an intuitive visualization of the model's complete computational graph, (3) it contains a built-in validation procedure to algorithmically verify the accuracy of all saved activations, and (4) it can be applied to arbitrary PyTorch models with no modifications, including models with dynamic computational graphs (e.g., if-then logic in the forward pass), recurrent models, models containing parallel branching, and models with internally generated tensors (e.g., injections of noise). Furthermore, using *TorchLens* requires minimal additional code, making it easy to incorporate into existing pipelines for model development and analysis, and useful as a pedagogical aid when teaching deep learning concepts.

Keywords: TorchLens; deep learning; PyTorch; artificial neural networks; Python

Introduction

Extracting hidden layer activations from deep neural networks (DNNs) is an essential step in understanding the series of transformations with which they transform their inputs into outputs, and for relating their intermediate processing stages to those used by the brain. Since it is increasingly common to compare many different DNNs based on their task performance and correspondence with the brain (Khaligh-Razavi & Kriegeskorte, 2014; Xu & Vaziri-Pashkam, 2021; Yamins et al., 2014), it is highly desirable to have efficient and flexible methods for extracting

intermediate activations from DNNs. Ideally, such a method should work for all PyTorch DNN models (not just a subset), should be able to extract the results of any desired intermediate operations without limitations, should make it easy to understand the placement of each layer within the broader network, and given the infinite space of possible DNNs, should have built-in methods for ensuring the accuracy of saved activations. While several PyTorch feature extraction packages exist (Marcel & Rodriguez, 2010; Muttenthaler & Hebart, 2021; Schneider, 2022), none meet all of these criteria: for instance, some only work for models with static computational graphs but not for models with dynamic graphs (e.g., from conditional if-then branching in the model's forward pass, or recurrent models with varying numbers of loops), and others can extract the results of PyTorch module objects, but not from tensor operations that are not linked to a module.

Here, we introduce a new Python package, *TorchLens*, that meets these criteria: it works for arbitrary PyTorch models, can extract the results of any desired tensor operation in a model, and can automatically visualize the structure of a network, aiding in layer selection and understanding the structure of a network. Finally, it has a built-in validation procedure for verifying the accuracy of saved activations, ensuring its robustness for novel architectures. We envision *TorchLens* being useful not only for streamlining analysis pipelines, but also for model prototyping and visualization, and as a pedagogical tool for teaching deep learning concepts.

Implementation and User Interface

TorchLens extracts the results of intermediate DNN operations by transiently decorating all PyTorch functions that return a tensor such that information about the inputs and outputs of that function call are logged. Unlike feature extraction approaches involving attaching forward hooks to PyTorch modules, this approach can save the results of operations that are not linked to PyTorch modules, and unlike approaches



1485–1488. <https://doi.org/10.1145/1873951.1874254>

Muttenthaler, L., & Hebart, M. N. (2021). THINGSvision: A Python Toolbox for Streamlining the Extraction of Activations From Deep Neural Networks. *Frontiers in Neuroinformatics*, 15. <https://www.frontiersin.org/articles/10.3389/fninf.2021.679838>

Khaligh-Razavi, S.-M., & Kriegeskorte, N. (2014). Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation. *PLOS Computational Biology*, 10(11), e1003915. <https://doi.org/10.1371/journal.pcbi.1003915> Placeholder Placeholder

Schneider, F. (2022). Surgeon-pytorch. <https://github.com/archinetai/surgeon-pytorch>

Xu, Y., & Vaziri-Pashkam, M. (2021). Limits to visual representational correspondence between convolutional neural networks and the human brain. *Nature Communications*, 12(1), Article 1. <https://doi.org/10.1038/s41467-021-22244-7>

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23), 8619–8624. <https://doi.org/10.1073/pnas.1403112111>