iNNvestigate-GUI - Explaining neural networks through an interactive visualization tool

iNNvestigate-GUI - Explaining Neural Networks Through an Interactive Visualization Tool

Fabio Garcea[1][0000−0003−3460−5297], Sina Famouri[1][000−0002−1164−6363], Davide Valentino[1], Lia Morra[1][0000−0003−2122−7178], and Fabrizio Lamberti[1][0000−0001−7703−1372]

Politecnico di Torino, Dipartimento di Automatica e Informatica, Turin, Italy
{fabio.garcea, sina.famouri, lia.morra, fabrizio.lamberti}@polito.it, davide.valentino@live.it

Abstract. In recent years, deep neural networks have reached state of the art performance across many different domains. Computer vision in particular has benefited immensely from deep learning. Despite their high performance, deep neural networks often lack interpretability and are mostly regarded as a black box. Therefore, the availability of tools capable to provide insights into the models and identify potential errors is crucial. Such tools need to seamlessly integrate within the workflow of data scientists and ML researchers. In this paper we propose iNNvestigate-GUI, an open-source graphical toolbox which offers an extensive set of functionalities for users to compare different networks behavior and give an explanation to their outputs.

Keywords: deep learning · convolutional neural networks · visualization · explainable artificial intelligence

1 Introduction

Deep learning has become a fundamental tool for a variety of applications. Thanks to easy-to-use libraries like Keras [5], a deep neural network (DNN) can be implemented in few lines of code, and many practitioners from a variety of fields can use DNNs despite limited knowledge and background in machine learning. However, practitioners still need crucial insights into the trained models. Typical examples that occur within the development life cycle include debugging models that do not converge or perform poorly on the target labels, or finding samples which the model cannot handle correctly for the specified task.

Given that DNNs are large models with millions of parameters trained on thousands of data points, visual analytics is emerging as a powerful tool to aid the inspection of DNNs and tackle the amount of data generated during their training [6, 13]. For instance, tools as Tensorboard allow to visualize gradients, activations, losses, etc. focusing on the training and optimization process.

Another important aspect of training DNNs is having insights into their properties, e.g., determining the most important features for classification. To

compensate for the black-box nature of DNNs, many eXplainable Artificial Intelligence (XAI) techniques have been designed to provide post hoc explanations of predictions. Techniques and algorithms that can provide visual interpretations are particularly effective, especially for DNNs targeting image interpretation [13]. Yet, there is a lack of tools to easily integrate such techniques in the development cycle. Activis [8] is one of the most comprehensive GUIs (Graphical User Interfaces) available for this purpose, but unfortunately, it is not publicly available. We argue that similar open source and publicly available interfaces are crucial to support the integration of XAI techniques in the development of deep learning models.

In this paper, we present iNNvestigate-GUI[1], an open-source, user-friendly, visual analytics tool for DNNs, especially tailored to computer vision. It is built upon the open-source iNNvestigate library [2], which provides a reference implementation of several visualization algorithms. Our aim is to provide a GUI which simplifies model interpretation by providing easy, code-free access to a comprehensive pool of visualization methods.

Designing such software has its challenges. First of all, visual comparison of several DNNs will be computationally expensive. The target users are diverse, with varying levels of machine learning knowledge and needs, so visual interpretability is a key factor. The tool should be easily integrated in the research and development workflow. Last but not least, as a crucial step towards XAI, when comparing DNNs one should not only take into account the final performance but also the reason behind the outputs.

We will explain how iNNvestigate-GUI fits into the literature in Section 2. In Section 3, the design challenges are explained in detail. Section 4 gives an overview of implementation details and design goals. The tool was tested with a group of non-expert users (computer science students). As described in Section 5, it reached a usability score of 73.34 according to the SUS scale.

2 Related Work

In this section we will first describe the currently available GUI-based tools and libraries to perform visual analysis in deep learning applications. Previous works have been categorized according to their license, availability and target audience, as detailed in Table 1.

2.1 Visual Analytic Tools for Deep Learning

Open-source tools for experienced users. This group covers the majority of visual analytic tools in literature and collects most of the most popular applications used for deep learning applications. The well known TensorFlow Graph Visualizer [21] for instance, which is part of the widely adopted TensorFlow framework, allows visualizing a neural network as a directed graph embedding other

[1] code is available at https://gitlab.com/grains2/innvestigate-gui/

crucial information like layers hyperparameters in a single scalable view. Embedding Projector [17] is another recent visualization tool developed by Google and included in the TensorFlow framework that allows plotting tensors in space through different dimensionality reduction techniques. Chung et al. proposed a dynamic real-time visual system to monitor the 2D representation of the filters learned by different layers and an interactive approach to steer the model configuration during the training process [4]. The Deep Visualization Toolbox [22] provides a matrix-like grid-view representation of the activations of the neurons in a given layer for a specific input image or video. A similar approach has been recently adopted in Summit [7], a tool developed to let practitioners and experts visualize neuron activations, thus enhancing the interpretability of the models. Several visualization tools targeting neural networks focus on the training and optimization phase. For instance, DeepEyes [11] supports the interpretation of the features learned by a CNN model during the training phase. As evident from Table 1, most of these tools allow visualization of gradients and activations and are more suited to effectively monitoring the training process than towards XAI.

Other recent visualization tools like LSTMvis [18] and GANviz [19] are instead focused on specific types of networks (such as LSTM or GANs), whereas in this work we are aiming at a more general purpose tool.

Proprietary tools for experienced users. ActiVis [8] is an example of proprietary Web application developed by Facebook that represents a comprehensive alternative to TensorFlow Graph Visualizer. It allows visualizing a neural network through a node-graph representation and performing behavioral analysis at different levels, from a subset of samples to a single instance and down to the activation of a single neuron. However, it is deployed on FBLearner Flow, the machine learning platform of Facebook, and is available only for internal researchers and practitioners. In [9], Shixia Liu et al. proposed a tool named CNNVis, which allows to visualize a clustered representation of the features learned by neurons and the connections between neurons at different layers with a minimal representation aimed at reducing the visual clutter caused by a high number of links between nodes. This tool has however no public implementation and only an online demo is currently available.

Educational tools. Some visualization tools are designed to help students better understand how neural networks work and, more in general, to be used for educational purposes [16, 20]. An an example, TensorFlow Playground is a web application developed by Google researchers, that allows manipulating interactively a simple model, including the structure, hyper-parameters and data points, to appreciate directly their effect on the decision boundaries learnt by the network. However, such tools are not adequate to visualize complex networks and datasets typical of real-life projects, even for inexperienced researchers.

2.2 Libraries of Visualization Algorithms

DNNs are generally regarded as black boxes due to their lack of explicit interpretability. To tackle this issue, several visualization algorithms have been

Table 1. Summary of the main deep learning visualization tools and comparison with the proposed tool.

	TF Graph Visualizer	**Embedding Projector**	**ActiVis**	**DeepVis**	**CNNVis**	**ReVACNN**	**DeepEyes**	**Summit**	**iNNvestigate-GUI**
Visualization									
Node-Link Graph	✓		✓						
Embeddings		✓	✓			✓	✓	✓	
Activations			✓	✓	✓	✓	✓	✓	✓
Gradients			✓	✓	✓	✓			
Hyperparameters	✓					✓		✓	
Attributions								✓	✓
Training History							✓		
Framework									
TensorFlow	✓	✓							✓
Keras	✓					✓			✓
FBlearner Flow			✓						
ConvNetJS						✓			
Caffe				✓			✓		
User Interface									
GUI (web-app)	✓	✓	✓			✓		✓	✓
GUI				✓			✓		
Command Line									
Availability									
Open Source	✓	✓		✓		✓	✓	✓	✓
Proprietary			✓						

proposed to help understanding why a model is producing a certain output for a given input [13]. These techniques visualize aspects such as the filters learned by a specific layer of the network, the activation of a certain neuron, or the gradients flowing through the layers. Perturbation-based methods stimulate and visualize changing network behavior by perturbing the input of the model. These methods rely on different visual paradigms varying from heatmaps to pixel display grids. Since the seminal work by Zeigler & Fergus [10], the number of available visualization techniques has been increasing steadily given the growing interest in XAI. A complete review is outside of the scope of this paper, and the reader is referred to many excellent surveys available [13, 6].

In practice, the applicability of visualization techniques is often hindered by a lack of publicly available reference implementations [2]. A few libraries have been recently proposed to gather and unify different visualization techniques in a common framework, including Keras Explain [1], DeepExplain [3] and iNNvestigate [2]. As detailed in Table 2 all include a variety of gradient-based (like DeepLIFT [15]), model-independent methods (like LIME [12]) and perturbation-based methods. Still, their integration in the model development cycle can be greatly simplified by providing a graphical interface, and presents several challenges which are discussed in Section 3.

3 Design Challenges

A series of joint design sessions were conducted by involving researchers and practitioners with different levels of expertise. Moving from the analysis of existing tools, illustrated in Section 2, we highlighted several critical gaps to be

Table 2. Summary of the available libraries of visualization algorithms.

	Keras Explain	**DeepExplain**	**iNNvestigate**
Visualization Technique			
Gradient/Saliency maps	✓	✓	✓
SmoothGrad			✓
DeconvNet			✓
Guided Backpropagation	✓		✓
PatternNet			
GradCAM	✓		
Guided GradCAM	✓		✓
Input * Gradient		✓	✓
LRP	✓	✓	✓
Integrated Gradients	✓	✓	✓
DeepTaylor			✓
DeepLIFT		✓	✓
Pattern Attribution			
Prediction Difference	✓		
Grey-box Occlusion	✓	✓	
LIME	✓		
Shapley Value Sampling		✓	

addressed, with an emphasis on open-source solutions. A seconded set of design challenges (C1-C5) was identified.

C1. Resource demanding visualizations. Training and evaluating DNNs is computationally expensive, especially when working with images. The time required to produce the visualizations should be limited in order to enhance user acceptability, but many of the existing visualization techniques are computationally intensive. Likewise, large datasets are usually involved in running deep learning experiments. Targeting the open-source community, the proposed implementation should allow a variety of computing setups, to access the provided visualization techniques.

C2. Diversified set of users. Designing a tool that is easy to use and accommodates both expert and non-expert users is challenging. Many of the available tools either target very inexperienced users and are mostly intended as teaching aids, or are designed to work in an industrial R&D environment where users are likely to have similar experience levels. Open-source tools target users who may have different needs and preferences. The workflow should follow a clear and simple structure; the different views of the interface should be self-explanatory and the visualizations designed with clarity, yet being capable of producing useful insights in non trivial projects.

C3. Performing instance-based and dataset-based analysis. Several XAI techniques are designed to provide a post hoc explanation of the predictions on a specific instance. However, as mentioned before, DNNs are trained and tested on very large datasets, and it is impractical for the user to manually comb through the dataset to find critical samples. A suggestion system is needed to rapidly identify data instances that are worthy of inspection.

C4. Simplify integration in R&D. The practical adoption of XAI techniques is often hindered by i) the lack of a reference, readily-available implementation and ii) the need to design and implement specific code for their

integration in the model development pipeline. A graphical tool should allow to produce the expected results significantly faster than writing code from scratch and, in general, generate the required visualization through a limited number of clicks.

C5. Model complexity and variety. Many visual analytics tools are designed to evaluate a single model, and often assume a relatively simple architecture. In practice, dozens of different models may need to be trained and compared, and we argue that this comparison should take into account not only performance but also the quality of the prediction and the presence of systematic biases [14].

4 Implementation

In this section, a detailed description of the functionalities offered by the iNNvestigate-GUI visualization tool is reported. In Section 4.1, the main design goals (G1 - G4) are described and motivated. Then, we move on to illustrate how these goals were achieved by designing a workflow for easy visualization of DNNs (Section 4.2) and for navigating a large dataset for sample selection (Section 4.3).

4.1 Design Goals

G1. Offering to researchers and practitioners a fast code-free tool for interpreting their models. Available visualization libraries represent different attempts to create a common reference implementation to tackle models explainability and interpretability [2]. Their use, however, passes through an Application Programming Interfaces (API); hence, time is needed to read the documentations and write ad hoc code. A visual analytics tool offers a ready-to-use common GUI to multiple visualization methods. It has to support inspections at different levels of depth, from the entire model to single layers and units.

G2. Easy graphical comparison of multiple models. It is common during a deep learning project to train and evaluate multiple models with different architectures, hyper-parameters and configurations. Since there is no consensus as to which visualization methods have the most desirable properties [2], the visualization tool must provide an easy interface to compare the pool of XAI techniques on multiple models.

G3. Allowing navigation of large scale datasets and identification of poorly classified and borderline data instances. Deep learning models are in general trained and validated on large scale datasets, whereas most visualization methods (with the notable exception of embeddings) operate at the instance level. Selecting interesting input instances for analysis is not straightforward. Random sampling is time consuming and may be lead to missed errors. The proposed tool must provide an intuitive and effective way to select samples that are worthy of further analysis, e.g., instances that the DNNs cannot classify correctly. This approach could both save the time

needed to perform an ad hoc analysis of the samples, and highlight crucial instances that may remain unnoticed, eventually increasing the capability of the tool to provide insights on the model behavior.

G4. Web-based implementation to tackle computationally demanding tasks. Although visualization is less computationally intensive than training, some visualization techniques still require an ad hoc training phase and indeed, a large number of samples may need to be processed. To empower users with different resources requirements and availability in terms of memory, disk space and computational power, we chose to develop the tool as a web application. The advantage of this approach is the capability to demand computationally demanding tasks to a back-end, possibly equipped with GPUs, while providing to the users a lightweight front-end accessible from anywhere through a web browser. This framework is already adopted by many popular tools (e.g., Tensorboard, Embedding Projector), and accommodates both users equipped with high-end workstations as well as those exploiting cloud computing services (e.g., Amazon Web Services).

4.2 Explaining Custom Models Through Visualization Methods

In this subsection, we describe the complete process to explain the behavior of DNNs through one or more visualization methods. Based on the analysis in Section 2, we selected the iNNvestigate [2] package as the reference implementation, and provided an ad hoc implementation for methods not included in this library, i.e., GradCAM and Guided GradCAM [14]. In addition, it is possible to visualize the output activations of a given neuron. These methods were included because they can produce particularly intuitive and easy-to-interpret visualizations especially suited to novice and non-expert users.

The iNNvestigate-GUI workflow starts by uploading the dataset and the pre-trained model(s). It is possible to analyze both models trained by the user or directly select the ImageNet-trained models available in the Keras library, which could be useful also for teaching purposes. For the visualization methods, all the options and all the required configurations parameters are provided as scroll-down lists to enhance the intuitiveness of the GUI. The tool also allows to specify a single layer or a single neuron to visualize the activation.

Once the setup is completed, the selected visualizations are generated and displayed to understand the DNNs behavior. The visualization panel is divided in multiple boxes, one for each of the models loaded in the configuration phase. The visualization method is applied to the output of the selected layer (the last convolutional layer by default) or neuron, for each data instance. Through an interactive panel it is thus possible to inspect the produced visualizations in a synchronized fashion, allowing fast and intuitive comparison between the behavior of multiple networks. For each data sample and DNN, the top predictions and their scores are shown next to the visualization output (see Fig. 1).

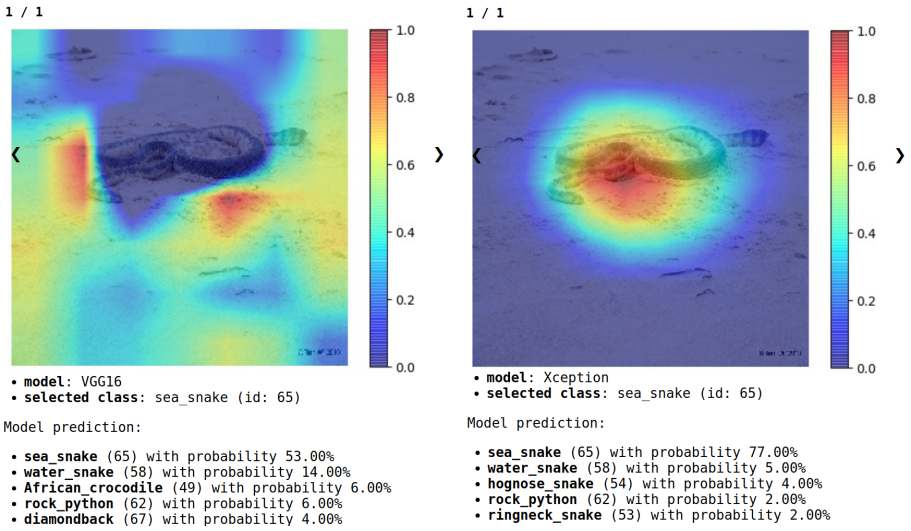

Fig. 1. Comparison of different model predictions for one of the images included in T1. The two models exploit different visual features to make the classification. The overlapping heatmaps have been produced using the GradCAM technique.

4.3 Suggesting Useful Data Samples for Analysis

iNNvestigate-GUI allows the user to easily identify useful samples to analyze thanks to the Suggestion panel (see Fig. 2). We assumed that the users should analyze the predictions for a mix of data instances with different properties: for instance, incorrectly classified samples allow the user to investigate the source of possible errors. Moving from these observations, the Suggestion panel categorizes available samples in order to allow the user to select a mix of samples with different properties for inspection. We identified two operating modalities based on i) whether a single or multiple models are compared and ii) whether the ground truth labels are available.

In the *multiple model* setting, the Suggestion panel shows a scatter plot of the input samples according to the confidence and agreement of the different models, as reported in Fig. 2. The mean prediction score across all models is reported on the x axis, and the number of predicted classes on the y axis. While hovering with the mouse over one of the data points in the scatter plot it is possible to have a visual preview of the examples. Based on their position in the chart different types of data instances can be distinguished.

Samples in the right-lower corner are images that are classified in the same way by all the evaluated models with a high prediction score. Samples in this are are classified in the same class with high confidence by all the models, and thus are likely to be correctly classified. Still, they could be interesting to inspect in order to exclude the presence of systematic biases in the dataset.

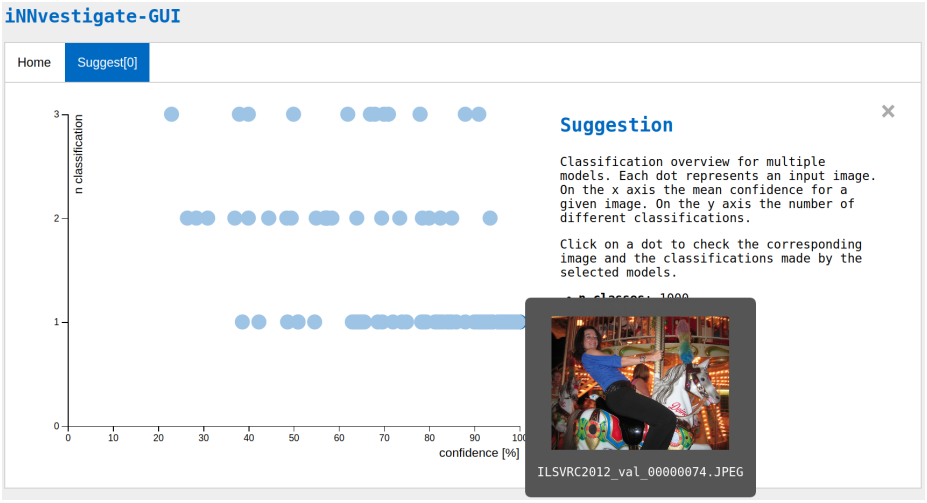

Fig. 2. *Suggestion* panel of iNNvestigate-GUI. A scatter plot represents the input dataset processed by multiple neural networks and guides the user towards selecting meaningful samples for further analysis. The average prediction score (x axis) is plotted against the number of different classes (y axis) predicted by the models. This visualization allows to identify data samples with high/low agreement among different models, as well as those predicted with high/low confidence.

Samples in the top-left corner (low agreement / low confidence) are probably borderline cases, or correctly classified by only a subset of the DNNs.

Samples in the top-right corner are predicted in different classes (low agreement) but with high prediction scores. They could include out-of- distribution samples on which DNNs are likely to misbehave, samples that may easily fool one or more of the models (including adversarial samples), or again, samples correctly classified only by a selection of models. Under this structure, users could focus their attention on the top-left and top-right quadrants.

In the *single model* setting, the Suggestion panel shows a simpler histogram plot. The user can choose to visualize the distribution of the activations for a pre-defined layer, e.g., to select the images that mostly excite a specific layer. Alternatively, for of labelled data, it is possible to plot the distribution of the difference between the prediction and the correct label (typically 1.0 for classification models) to identify samples that are correctly or incorrectly classified.

5 A Usability Test Case

To better evaluate the usability of the tool, two tasks have been prepared and submitted to a group of 9 computer engineering students (with a previous background in deep learning) at Politecnico di Torino. All students had at least a

basic knowledge of CNNs and attended at least one course in machine learning. A set of questions was prepared to guide the users through the completion of the two tasks. After completing the tasks, all users were administered a questionnaire according to the SUS (System Usability Scale) approach. The two tasks are summarized as follows:

T1. Evaluate the visual features used by different models to predict the same subset of input images. This task emulates the comparison of different trained models. A subset of 100 data samples from the ImageNet ILSVRC2012 dataset was selected for this task.

T2. Assess whether multiple models use appropriate visual features to classify a subset of inputs of the same class. This task emulates the search for visual biases. For instance, a network may inadvertently learn to predict an object based on co-occurring background features. For this task a subset of 15 images belonging to the *golden_retriever* class was randomly selected from the ImageNet ILSVRC2012 dataset.

The users had to compare three popular models available in Keras (VGG16, ResNet50 and Xception) with varying level of complexity. In order to reduce the time needed to complete the task, available visualization algorithms where restricted to Gradient/Saliency, Guided Backpropagation, GradCAM e LRP-z.

During task T1, all users could successfully use the Suggestion panel to identify images where the models agreed/disagreed or had low/high prediction confidence. In particular, users focused their search in the right-lower quadrant (high agreement/high confidence, see Fig.3) and left-top quadrant (low agreement/low confidence). In both cases, users found GradCAM to be the easiest method to interpret (66.7% and 55.6% of the users, i.e. 6 out of 9 and 5 out of 9 respectively).

In task T2, the participants were asked to identify samples that were classified correctly by all the models and samples with inconsistent behavior. Results showed that 77.8% (7/9) of the participants relied on the histogram chart to identify both, while 22.2% of the users (2/9) also relied on the scatter plot. In this case the most intuitive algorithm for the analysis of visual features of images classified with the correct labels by most of the models was Guided Backpropagation (55.6% of the votes, i.e. 5/9). On the other side, when analyzing images that were incorrectly classified, the most popular choice was again GradCAM (66.7% of users, i.e. 6/9).

In task T2, users were asked to rate the visual explanations on a scale from 1 (completely agreeing) to 5 (completely disagreeing). 55.6% (5/9) of the participants was completely satisfied by the visual explanations for the VGG16 and ResNet50 models (meaning they found the proposed attributions appropriate for the label), whereas only 44.4% (4/9) were satisfied with the Xception network.

The mean usability score was 73.34%, which according to the SUS usability scale is above average (any value above 68% is considered above the average).

Moreover, 77.8% (7/9) of the participants declared that they would have not been able to easily solve both tasks without iNNvestigate-GUI, while only 22.2% (2/9) stated they could solve the same tasks writing ad hoc code.

Fig. 3. Visual comparison of the predictions made by the three models in T1. The overlapping heatmap has been produced by the GradCAM algorithm. This is image was selected by the majority of the participants from the *Suggestion* view as an example of high confidence and high agreement classification. All the models are focusing on similar visual features to classify the frame. Best seen in RGB; consider brighter areas of the frames in case of gray scale visualization.

6 Conclusion

This paper proposed a new GUI-based Web application featuring a comprehensive pool of XAI visualization algorithms, and intended to compare and understand multiple DNN models. The tool leverages an existing open-source library, named iNNvestigate, for existing implementation of visualization algorithms.

In contrast to other existing tools, the proposed GUI allows a fast comparison between multiple models at different levels of depth and implements a suggestion strategy to highlight the most critical data samples. The target users for the tool span from the inexperienced learner to researchers and deep learning practitioners. As demonstrated by preliminary user experiments, it can be exploited by inexperienced users to improve and speed up DNNs development.

We plan to extend iNNvestigate-GUI by adding support for deep learning frameworks other than Keras and implementing additional visualizations, such as node-link diagrams, to simplify the selection and inspection of individual layers. More complex use cases are needed to demonstrate how visual analytics can prevent biases and errors to be introduced during model training.

References

1. Keras Explain (2018), https://github.com/primozgodec/keras-explain
2. Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K.T., Montavon, G., Samek, W., Müller, K.R., Dähne, S., Kindermans, P.J.: iNNvestigate neural networks! J. Mach. Learn. Res. **20**(93), 1–8 (2019)
3. Ancona, M.: DeepExplain (2017), https://github.com/marcoancona/DeepExplain
4. Chung, S., Suh, S., Park, C., Kang, K., Choo, J., Kwon, B.C.: Revacnn: Real-time visual analytics for convolutional neural network. In: KDD 16 Workshop on Interactive Data Exploration and Analytics. pp. 30–36 (2016)
5. Gulli, A., Pal, S.: Deep learning with Keras. Packt Publishing Ltd (2017)

6. Hohman, F., Kahng, M., Pienta, R., Chau, D.H.: Visual analytics in deep learning: An interrogative survey for the next frontiers. IEEE transactions on visualization and computer graphics **25**(8), 2674–2693 (2018)
7. Hohman, F., Park, H., Robinson, C., Chau, D.H.P.: S ummit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. IEEE transactions on visualization and computer graphics **26**(1), 1096–1106 (2019)
8. Kahng, M., Andrews, P.Y., Kalro, A., Chau, D.H.P.: Activis: Visual exploration of industry-scale deep neural network models. IEEE transactions on visualization and computer graphics **24**(1), 88–97 (2017)
9. Liu, M., Shi, J., Li, Z., Li, C., Zhu, J., Liu, S.: Towards better analysis of deep convolutional neural networks. IEEE transactions on visualization and computer graphics **23**(1), 91–100 (2016)
10. Matthew, D., Fergus, R.: Visualizing and understanding convolutional neural networks. In: Proceedings of the 13th European Conference Computer Vision and Pattern Recognition, Zurich, Switzerland. pp. 6–12 (2014)
11. Pezzotti, N., Höllt, T., Van Gemert, J., Lelieveldt, B.P., Eisemann, E., Vilanova, A.: Deepeyes: Progressive visual analytics for designing deep neural networks. IEEE transactions on visualization and computer graphics **24**(1), 98–108 (2017)
12. Ribeiro, M.T., Singh, S., Guestrin, C.: " 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. pp. 1135–1144 (2016)
13. Seifert, C., Aamir, A., Balagopalan, A., Jain, D., Sharma, A., Grottel, S., Gumhold, S.: Visualizations of deep neural networks in computer vision: A survey. In: Transparent Data Mining for Big and Small Data, pp. 123–144. Springer (2017)
14. Selvaraju, R.R., Cogswell, M., Das, A., et al.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proc. of the IEEE International Conference on Computer Vision. pp. 618–626 (2017)
15. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 3145–3153. JMLR. org (2017)
16. Smilkov, D., Carter, S., Sculley, D., Viégas, F.B., Wattenberg, M.: Direct-manipulation visualization of deep networks. arXiv:1708.03788 (2017)
17. Smilkov, D., Thorat, N., Nicholson, C., Reif, E., Viégas, F.B., Wattenberg, M.: Embedding projector: Interactive visualization and interpretation of embeddings (2016)
18. Strobelt, H., Gehrmann, S., Pfister, H., Rush, A.M.: Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks (2016)
19. Wang, J., Gou, L., Yang, H., Shen, H.W.: Ganviz: A visual analytics approach to understand the adversarial game. IEEE transactions on visualization and computer graphics **24**(6), 1905–1917 (2018)
20. Wang, Z.J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., Kahng, M., Chau, D.H.: Cnn explainer: Learning convolutional neural networks with interactive visualization. arXiv preprint arXiv:2004.15004 (2020)
21. Wongsuphasawat, K., Smilkov, D., Wexler, J., Wilson, J., Mane, D., Fritz, D., Krishnan, D., Viégas, F.B., Wattenberg, M.: Visualizing dataflow graphs of deep learning models in tensorflow. IEEE transactions on visualization and computer graphics **24**(1), 1–12 (2017)
22. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579 (2015)