TorchEsegeta: Framework for Interpretability and Explainability of Image-Based Deep Learning Models

Soumick Chatterjee [1,2,3,*], Arnab Das [1], Chirag Mandal [1], Budhaditya Mukhopadhyay [1], Manish Vipinraj [1], Aniruddh Shukla [1], Rajatha Nagaraja Rao [1], Chompunuch Sarasaen [3,4], Oliver Speck [3,5,6,7] and Andreas Nürnberger [1,2,6]

[1] Faculty of Computer Science, Otto von Guericke University Magdeburg, 39106 Magdeburg, Germany; arnab.das@st.ovgu.de (A.D.); chirag.mandal@st.ovgu.de (C.M.); budhaditya.mukhopadhyay@st.ovgu.de (B.M.); manish.vipinraj@ovgu.de (M.V.); aniruddh.shukla@st.ovgu.de (A.S.); ohne.rajathanagaraja@ovgu.de (R.N.R.); andreas.nuernberger@ovgu.de (A.N.)

[2] Data and Knowledge Engineering Group, Otto von Guericke University Magdeburg, 39106 Magdeburg, Germany

[3] Biomedical Magnetic Resonance, Otto von Guericke University Magdeburg, 39106 Magdeburg, Germany; chompunuch.sarasaen@ovgu.de (C.S.); oliver.speck@ovgu.de (O.S.)

[4] Institute for Medical Engineering, Otto von Guericke University Magdeburg, 39106 Magdeburg, Germany

[5] German Center for Neurodegenerative Disease, 39120 Magdeburg, Germany

[6] Center for Behavioral Brain Sciences, 39106 Magdeburg, Germany

[7] Leibniz Institute for Neurobiology, 39118 Magdeburg, Germany

[*] Correspondence: soumick.chatterjee@ovgu.de

Abstract: Clinicians are often very sceptical about applying automatic image processing approaches, especially deep learning-based methods, in practice. One main reason for this is the black-box nature of these approaches and the inherent problem of missing insights of the automatically derived decisions. In order to increase trust in these methods, this paper presents approaches that help to interpret and explain the results of deep learning algorithms by depicting the anatomical areas that influence the decision of the algorithm most. Moreover, this research presents a unified framework, TorchEsegeta, for applying various interpretability and explainability techniques for deep learning models and generates visual interpretations and explanations for clinicians to corroborate their clinical findings. In addition, this will aid in gaining confidence in such methods. The framework builds on existing interpretability and explainability techniques that are currently focusing on classification models, extending them to segmentation tasks. In addition, these methods have been adapted to 3D models for volumetric analysis. The proposed framework provides methods to quantitatively compare visual explanations using infidelity and sensitivity metrics. This framework can be used by data scientists to perform post hoc interpretations and explanations of their models, develop more explainable tools, and present the findings to clinicians to increase their faith in such models. The proposed framework was evaluated based on a use case scenario of vessel segmentation models trained on Time-of-Flight (TOF) Magnetic Resonance Angiogram (MRA) images of the human brain. Quantitative and qualitative results of a comparative study of different models and interpretability methods are presented. Furthermore, this paper provides an extensive overview of several existing interpretability and explainability methods.

Keywords: deep learning; black box; interpretability; explainability; model introspection; MRA segmentation

1. Introduction

The use of artificial intelligence is widely prevalent today in medical image analysis. However, it is imperative to do away with the black-box nature of deep learning techniques to gain the trust of radiologists and clinicians, as a model's erroneous output might have a high impact in the medical domain.

Interpretability vs. Explainability: Interpretability and explainability methods aim to unravel the black-box nature of decision making in machine learning and deep learning models. Explainability focuses on explaining the internal working mechanisms of the model. In contrast, interpretability focuses on observing the effects of changes in model parameters and inputs on the model prediction, hence attributing properties to the input, and it also requires a context, considering the audience who is to interpret the properties and characteristics for the model outcomes [1,2]. It is to be noted that earlier work failed to draw a clear line of distinction between interpretability and explainability as these terms are subjective and pertain to the stakeholders, considering the audience who must understand the model and its outcomes. Interpretability and explainability nurture a sense of human–machine trust as they help the users of the machine/deep-learning models understand how certain decisions are made by the model and are not limited to statistical metric-based approaches such as accuracy or precision. In turn, this allows employing such models in mission-critical problems such as medicine, autonomous driving, legal systems, banking, etc. The basis of measuring model transparency is said to comprise simulatability, decomposability, and algorithmic transparency [2]; and model functionality, which consists of textual descriptions of the model's output and visualisations of the model's parameters. The goals to be attained through the interpretability of models [1] are trust, reliability, robustness, fairness, privacy, and causality. Explainability can be formulated as the explanation of the decisions made internally by the model that in turn generates the observable, external conclusions arrived at by the model. This promotes human understanding of the model's internal mechanisms and rationale, which in turn helps build user trust in the model [3]. The evaluation criteria for model explainability include comprehensibility by humans, fidelity and scalability of the extracted representations, and scalability and generality of the method [4].

Some models are transparent and hence inherently understandable, such as Decision Trees, K-Nearest Neighbors, Rule-based, and Bayesian models. There are even some deep learning-based models, such as GP-U-Net [5] and CA-Net [6], which are inherently trying to provide an explanation concerning their outcome. but some are opaque and require post hoc explanations. A post hoc model explanation can be through model-agnostic and model-specific techniques that aim towards explaining a black-box model in human-understandable terms. The European Union has also incorporated the transparency and accountability of models in 2016 as a criterion in the ethical guidelines for trustworthy technology AI [7,8]. Through interpretability and explainability, to some degree, the following goals can be achieved: trustworthiness, causality, confidence, fairness, informativeness, transferability, and interactivity, which in turn can help improve the model [7].

Data scientists can generate interpretability–explainability results for their opaque models and present the model's reasoning or the model's inner mechanism to the domain experts (e.g., clinicians). If the reasoning is the same as a human domain expert would have done, then the experts can have more faith in that model, which in turn implies that the model can be incorporated into real-life workflows. Apart from building trust in the models, interpretability and explainability techniques can be used by the data scientists to improve their models as well, by discussing the outcomes with the domain experts and improving the model based on expert feedback. Moreover, accurate (verified by experts) interpretability–explainability results can be used as part of automatic or semi-automatic teaching programs for trainees.

For classification models, there are multiple interpretability and explainability techniques supported by various libraries such as Captum (https:captum.ai), Torchray [9], and CNN Visualisation library (https://github.com/utkuozbulak/pytorch-cnn-visualizations), and many of the techniques are introduced in the following section of this paper. However, this is more challenging for segmentation, as the output is more complex than just a simple class prediction. In this contribution, various interpretability and explainability techniques used for classification models have been adapted to work with segmentation models. On applying the interpretability techniques, essential features or areas of the input image can

be visualised, on which the model's output is critically based. By applying explainability techniques, a better understanding of the knowledge represented in the model's parameters can be achieved.

Contributions

This paper proposes a unified, flexible, and scalable interpretability–explainability pipeline for PyTorch, TorchEsegeta, which leverages post hoc interpretability and explainability methods and can be applied on 2D or 3D deep learning models working with images. Apart from implementing the existing methods for classification models, this research extends them for segmentation models. This pipeline can be applied to trained models with little to no modification, using either a graphical user interface for easy access or directly using a python script. It also provides an easy platform for incorporating new interpretability or explainability techniques. Moreover, this pipeline provides features to evaluate the interpretability and explainability methods using two different methods. First, using cascading randomisation of the model's weights, which can help evaluate how much the results are dependent on the actual weights, and second, using quantitative metrics: infidelity and sensitivity. Finally, the pipeline was applied on models trained to segment vessels from magnetic resonance angiograms (MRA), and the interpretability results are shown here. Furthermore, apart from the technical contributions with the TorchEsegeta pipeline, this paper also provides a comprehensive overview of several post hoc interpretability and explainability techniques.

2. Methods

This paper presents the TorchEsegeta framework, which integrates various interpretability and explainability techniques available in different libraries and extends these techniques for segmentation models. It is noteworthy that the development of this pipeline started with the exploration of various interpretability techniques for classifying COVID-19 and other types of pneumonia [10]. An initial pipeline was developed under that project for classification models but only for 2D images. This research further extends that for 3D volumetric images, as well as for segmentation models. Apart from incorporating these features, the original pipeline was further streamlined and improved during this research to create the first version of TorchEsegeta.

2.1. Incorporated Libraries

The following interpretability and explainability-based libraries have been explored in this research work— Captum, CNN visualisation, TorchRay, DeepDream, Lucent, LIME, and SHAP.

Captum is a library built on PyTorch and is used to provide the interpretability of machine/deep-learning models. It provides many algorithms that evaluate the contribution of different features in providing a model's prediction and thus helps improve the model.

CNN visualisation [11] provides different implementations for the different interpretability techniques and visualisations for CNN-based model architectures.

TorchRay (https://github.com/facebookresearch/TorchRay) is a package used for several visualisation methods for convolutional neural network architectures using PyTorch. It focuses on interpretability wherein it attempts to determine which regions of the input image influence the final prediction made by the model [9].

LIME (Local Interpretable Model-agnostic Explanations) is a model-agnostic technique that provides post hoc model explanations to explain the decisions of the deep learning model, and due to its model-agnostic nature, LIME flexibly explains any unknown model [7,8,12].

SHAP (https://christophm.github.io/interpretable-ml-book/shap.html) (SHapley Additive exPlanation) is a post hoc, game-theoretical approach that computes shapley values in order to explain the prediction made by the deep learning model [13].

Lucent (https://github.com/greentfrapp/lucent) is the PyTorch implementation of lucid for the explainability of deep learning models. It aims to explain the decision made

by the deep neural network by explaining what is being learnt by the various layers of the network.

DeepDream (https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html) is also an explainability technique that is used to visualise the parameters learnt by the convolutional neural network.

2.2. Implemented Interpretability Techniques

Interpretability techniques help to understand the focus area of a model and the reasoning done by the model. These techniques can be categorised into two groups: model attribution and layer attribution.

2.2.1. Model Attribution Techniques

Model attribution techniques are the techniques used to assess the contribution of each attribute to the prediction of the model. There is a long list of methods available in the literature under the rubric model attribution techniques; hence, for better understanding, they are further divided here into two groups: feature-based and gradient-based.

1. *Feature-Based Techniques:*
    Methods under this group bring into play input and/or output feature space to compute local or global model attributions.

    (a) *Feature Permutation*—This is a perturbation-based technique [14,15] in which the value of an input or group of inputs is changed, utilising the random permutation of values in a batch of inputs and calculating its corresponding change in output. Hence, meaningful feature attributions are calculated only when a batch of inputs is provided.

    (b) *Shapley Value Sampling*—The proposed method defines an input baseline to which all possible permutations of the input values are added one at a time, and the corresponding output values and hence each feature attribution is calculated. For permutation O, given a player set N, the set of all possible permutations $\pi(N)$, and the predecessors of player i $Pre^i(O)$, the Shapley value is given by

    $$Sh_i(v) = \sigma_{O \in \pi(N)} \frac{1}{n!} (v(Pre^i(O) \bigcup i) - v(Pre^i(O))). \tag{1}$$

    As all possible permutations are considered, this technique is computationally expensive, and this can be overcome by sampling the permutations and averaging their marginal contributions instead of considering all possible permutations such as the ApproShapley sampling technique [16].

    (c) *Feature Ablation*—This method works by replacing an input, or a group of inputs with another value defined by a range or reference value, and the feature attribution of the input or group of inputs such as a segment are computed. This method works based on perturbation.

    (d) *Occlusion*—Similar to the Feature Ablation method, Occlusion [17] works as a perturbation-based approach wherein the continuous inputs in a rectangular area are replaced by a value defined by a range or a reference value. Using this approach, the change in the corresponding outputs is calculated in order to find the attribution of the feature.

    (e) *RISE Randomised Input Sampling for Explanation of Black-box Models*—It generates an importance map indicating how salient each pixel is for the model's prediction [18]. RISE works on black-box models, since it does not consider gradients while making the computations. In this approach, the model's outputs are tested by masking the inputs randomly and calculating the importance of the features.

(f) *Extremal Perturbations*—They are regions of an image that maximally affect the activation of a certain neuron in a neural network for a given area in an image [9]. Extremal Perturbations lead to the largest change in the prediction of the deep neural network when compared to other perturbations defined by

$$m_a = \underset{m:||m||_1 a|\Omega|, m \in M}{\arg\max} \Phi(m \otimes x) \tag{2}$$

for the chosen area a. The paper also introduces area loss to enforce the restrictions while choosing the perturbations.

(g) *Score-Weighted Class Activation (Score CAM)*—It is a gradient-independent interpretability technique based on class activation mapping [19]. Activation maps are first extracted, and each activation then works as a mask on the original image, and its forward-passing score on the target class is obtained. Finally, the result can be generated by the linear combination of score-based weights and activation maps (https://github.com/haofanwang/Score-CAM). Given a convolutional layer l, class c, number of channels k, and activations A:

$$L^c_{Score-CAM} = ReLU(\sum_k \alpha^c_k A^k_l). \tag{3}$$

Gradient-Based Techniques

This group of methods mainly use the model's parameter space to generate the attribute maps, which are calculated with the help of the gradients.

(a) *Saliency*—It was initially designed for visualising the image classification done by convolutional networks and the saliency map for a specific image [20]. It is used to check the influence of each pixel of the input image in assigning a final class score to the image I using the linear score model $S_c(I) = w_c^T I + b_c$ for weight w and bias b.

(b) *Guided Backpropagation*—In this approach, during the backpropagation, the gradients of the outputs are computed with respect to the inputs [21]. The RELU activation function is applied to the input gradients, and direct backpropagation is performed, ensuring that the backpropagation of non-negative gradients does not occur.

(c) *Deconvolution*—This approach is similar to the guided backpropagation approach wherein it applies RELU to the output gradients instead of the input gradients and performs direct backpropagation [17]. Similarly, RELU backpropagation is overridden to allow only non-negative gradients to be backpropagated.

(d) *Input X Gradient*—It extends the saliency approach such that the contribution of each input to the final prediction is checked by multiplying the gradients of outputs and their corresponding inputs in a setting of a system of linear equations AX = B where A is the gradient and B is the calculated final contribution of input X [22].

(e) *Integrated Gradients*—This technique [23] calculates the path integral of the gradients along the straight line path from the baseline x' to the input x. It satisfies the axioms of Completeness i.e., the attributions must account for the difference in output for the baseline x' and input x, Sensitivity—i.e., a non-zero attribution must be provided even to inputs that flatten the prediction function where the input differs from the baseline—and Implementation Invariance of gradients—i.e., two functionally equivalent networks must have identical feature attributions. It requires no instrumentation of the deep neural network

for its application. All the gradients along the straight line path from x' to x are integrated along the *i*th dimension as:

$$IG_i(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha. \tag{4}$$

(f)　*Grad Times Image*—In this technique [22], the gradients are multiplied with the image itself. It is a predecessor of the DeepLift method, as the activation of each neuron for a given input is compared to its reference activation, and contribution scores are assigned according to the difference for each neuron.

(g)　*DeepLift*—It is a method [24] that considers not only the positive but also the negative contribution scores of each neuron based on its activation concerning its reference activation. The difference in the output of the activation function of the neuron t under observation is calculated based on the difference in input with respect to a reference input. Contribution scores for each of the preceding neurons $x_1, \ldots, x_n$ that influence t are assigned $C_{\triangle x_i \triangle t}$, which sums up to the difference in t's activation:

$$\sum_{i=1}^{n} C_{\triangle x_i \triangle t} = \triangle t. \tag{5}$$

(h)　*DeepLiftShap*—This method extends the DeepLift method and computes the SHAP values on an equivalent, linear approximation of the model [13]. It assumes the independence of input features. For model f, the effective linearization from SHAP for each component is computed as:

$$\phi_i(f_3, y) \approx m_{y_i f_3}(y_i - E[y_i]). \tag{6}$$

(i)　*GradientShap*—This technique also assumes the independence of the inputs and computes the game-theoretic SHAP values of the gradients on the linear approximation of the model [13]. Gaussian noise is added to randomly selected points, and the gradients of their corresponding outputs are computed.

(j)　*Guided GradCAM*—The Gradient-Weighted Class Activation Mapping approach provides a means to visualise the regions of an image input that are predominant in influencing the predictions made by the model. This is done by visualising the gradient information pertaining to a specific class in any layer of the user's choice. It can be applied to any CNN-based model architecture. The guided backpropagation and GradCAM approaches are combined by computing the element-wise product of guided backpropagation attributions with upsampled GradCAM attributions [25].

(k)　*Grad-Cam++*—Generalised Gradient-Based Visual Explanations for Deep Convolutional Networks is a method [26] that claims to provide better predictions than the Grad-CAM and other state-of-the-art approaches for object localisation and explaining occurrences of multiple object instances in a single image. This technique uses a weighted combination of the positive partial derivatives of the last convolutional layer's feature maps concerning a specific class score as weights to generate a visual explanation for the corresponding class label. The importance w_k^c of an activation map A^k over class score Y^c is given by

$$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc}.relu\left(\frac{\partial Y^c}{\partial A_{ij}^k}\right). \tag{7}$$

Grad-CAM++ provides human-interpretable visual explanations for a given CNN architecture across multiple tasks, including classification, image caption generation, and 3D action recognition.

(l)　　*Vanilla Backpropagation and Layer Visualisation*—It is the standard gradient back-propagation technique through the deep neural network wherein the gradients are visualised at different layers and what is being learnt by the model is observed, given a random input image.

(m)　　*Smooth Grad*—In this method [19,27], random Gaussian noise $N(0, \sigma^2)$ is added to the given input image, and the corresponding gradients are computed to find the average values over n image samples

$$\hat{M}_c(x) = \frac{1}{n} \sum_1^n M_c(x + N(0, \sigma^2))　\quad\quad (8)$$

Vanilla and guided backpropagation techniques can be used to calculate the gradients.

2.2.2. Layer Attribution Techniques

Layer attribution methods evaluate the effect of each individual neuron in a particular layer on the model output. There are various types of layer attribution methods that have been explored as part of this research work. Perturbation-based approaches [19] perturb the original input and observe the change in the prediction of the model, and they are highly time-consuming.

1.　*Inverted Representation*—The aim of this technique is to generate the given input image after a number of n target layers. An inverse of the given input image representation is computed [28] in order to find an image $\Phi(x_0)$ whose representation best matches the input image Φ_0 while minimising the given loss l such that

$$x^* = \underset{x \in \mathbb{R}^{HxWxC}}{\arg\min} \; l(\Phi(x), \Phi_0) + \lambda R(x).　\quad\quad (9)$$

2.　*Layer Activation with Guided Backpropagation*—This method [21] is quite similar to guided backpropagation, but instead of guiding the signal from the last layer and a specific target, it guides the signal from a specific layer and filter. The guided backpropagation method adds an additional guidance signal from the higher layers to the usual backpropagation.

3.　*Layer DeepLift*—This is the DeepLift method as mentioned in the model attribution techniques [13] but applied with respect to the particular hidden layer in question.

4.　*Layer DeepLiftShap*—This method is similar to the DeepLIFT SHAP technique mentioned in the model attribution techniques [13], which was applied for a particular layer. The original distribution of baselines is taken, the attribution for each input–baseline pair is calculated using the Layer DeepLIFT method, and the resulting attributions are averages per input example. Assuming model linearity, $\phi(f_3, y) \approx m_y i$.

5.　*Layer GradCam*—The GradCam attribution for a given layer is provided by this method [25]. The target output's gradients are computed concerning the particularly given layer. The resultant gradients are averaged for each output channel (dimension 2 of output). Then, the average gradient for each channel is multiplied by the layer activations. Then, the results are added over all the channels. For the class feature weights w, global average pooling is performed over the feature maps A such that

$$S^c = \sum_k w_k^c \frac{1}{Z} \sum_i \sum_j A_{ij}^k.　\quad\quad (10)$$

6.　*Layer GradientShap*—This is analogous with the GradientSHAP [13] method as mentioned in the model attribution techniques but applied for a particular layer. Layer GradientSHAP adds Gaussian noise to each input sample multiple times, wherein a random point along the path between the baseline and input is selected, and the gradient of the output with respect to the identified layer is computed. The final SHAP

values approximate the expected value of gradients $*$ (layer activation of inputs $-$ layer activation of baselines).

7.　*Layer Conductance*—This method [29] provides the conductance of all neurons of a particular hidden layer. Conductance of a particular hidden unit refers to the flow of Integrated Gradients attribution through this hidden unit. The main idea behind using this method is to decompose the computation of the Integrated Gradients via the chain rule. One property that this method of conductance satisfies is that of completeness. Completeness means that the conductances of a particular layer add up to the prediction difference of F(x) – F(x') for input x and baseline input x'. Other properties that are satisfied by this method are those of Linearity and insensitivity.

8.　*Internal Influence*—This method calculates the contribution of a layer to the final prediction of the model by integrating the gradients with respect to the particular layer under observation. The internal representation is influenced by an element *j* as defined by [30] such that

$$\chi_i^s(f, P) = \int_\chi \frac{\partial g}{\partial z_j}\bigg|_{h(x)} P(x)dx \tag{11}$$

wherein $s = \langle g, h \rangle$ for the slice of the network represented by s as a function of tuple g,h. It is similar to the integrated gradients approach where instead of the input, the gradients are integrated with respect to the layer.

9.　*Contrastive Excitation Backpropagation/Excitation Backpropagation*—This approach is used to generate and visualise task-specific attention maps. The Excitation Backprop method as proposed by [31] is to pass along top–down signals downwards in the network hierarchy via a probabilistic Winner-Take-All process wherein the most relevant neurons in the network are identified for a given top–down signal. Both top–down and bottom–up information is integrated to compute the winning probability of each neuron as defined by

$$y_i = \sum_{j=1}^{N} w_{ij}x_j. \tag{12}$$

for input *x* and weight matrix *w*. The contrastive excitation backpropagation is used to make the top–down attention maps more discriminative.

10.　*Layer Activation*—It computes the activation of a particular layer for a particular input image [32]. It helps to understand how a given layer reacts to an input image. One can get an excellent idea of at what part or features of the image a particular layer looks.

11.　*Linear Approximator*—This is a technique to overcome inconsistencies of post hoc model interpretation; linear approximator combines a piecewise linear component and a nonlinear component [9,33]. The piecewise linear component describes the explicit feature contributions by piecewise linear approximation that increases the expressiveness of the deep neural network. The nonlinear component uses a multilayer perceptron to capture feature interactions and implicit nonlinearity, which in turn increases the prediction performance. Here, the interpretability is obtained once the model is learned in the form of feature shapes and has high accuracy.

12.　*Layer Gradient X Activation*—This method [34] computes the element-wise product of a given layer's gradient and activation. It is a combination of the gradient and activation methods of layer attribution. The output attributions are returned as a tuple if the layer input/output contains multiple tensors or a single tensor is returned.

2.3. Implemented Explainability Techniques

Explainability techniques aim to unravel the internal working mechanism of a model; they try to explain the knowledge represented inside the model's parameters.

2.3.1. DeepDream

As the network is trained using many examples, it is essential to check what has been learnt from the input image. Hence, visualising what has been learnt at different layers of the CNN-based network gives rise to repetitive patterns of different levels of abstraction, enabling interpretation of what has been learnt by the layers. This visualisation can be realised using DeepDream. Given an arbitrary input image, any layer from the network can be picked, and the detected features at that layer can be enhanced. It is observed that the initial layers are sensitive to basic features in the input images, such as edges, and the deeper layers identify complex features from the input image. As an example, InceptionNet (https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html) was trained on animal images, and it was observed that the different layers of the network could interpret an interesting remix of the learnt animal features in any given image.

2.3.2. LIME

Local Interpretable Model-agnostic Explanations (LIME) is a technique for providing post hoc model explanations. LIME constructs an approximated surrogate, locally linear model or decision tree of a given complex model that helps to explain the decisions of the original model, and due to its model-agnostic nature, it can be employed to explain any model even when its architecture is unknown [7,8,12]. LIME also performs the transformation of input features to obtain a representation that is interpretable to humans [4]. In the original work [35], the authors propose LIME, which is an algorithm that provides explanations for a model f that are locally faithful in the locality Π_x, for an individual prediction of the model, such that users can ensure that they can trust the prediction before acting on it. LIME provides an explanation for f in the form of a model g and $g \in G$ where G is a set of all possible interpretable models and $\Omega(g)$, which is the complexity of the interpretable model, is minimised. $L(f, g, \Pi_x)$ is the fidelity function that measures the unfaithfulness of g in estimating f in the locality Π_x. Hence, the explanation provided by LIME is [35]:

$$\xi(x) = argmin_{g \in G} L(f, g, \Pi_x) + \Omega(g). \tag{13}$$

2.3.3. SHAP

SHapley Additive exPlanation [13] is similar to LIME such that it approximates an interpretable, explanation model g of the original, complex model f, in order to explain a prediction made by the model f(x). SHAP provides post hoc model explanations for an individual output and is model-agnostic. It calculates the contribution of each feature in producing the final prediction and is based on the principles of Game Theory such as Shapley Values [4]. SHAP works on simplified, interpretable input data x', which is analogous to the original input data such that $x = h_x(x)$. SHAP must ensure consistency theorems [13], local accuracy such that $g(x')$ matches $f(x)$ when $x = h_x(x)$, and missingness such that if an attribute's value is 0, it does not not imply that its corresponding contribution to the prediction is 0. Hence, the contribution of each attribute x can be calculated as follows [13]:

$$\phi_i(f, x) = \Sigma_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') f_x(z' \setminus i)] \tag{14}$$

where $z' \in \{0, 1\}^M$, $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

2.3.4. Lucent

Based on Lucid (https://github.com/tensorflow/lucid), Lucent is a library implemented using PyTorch for the explainability of deep learning models. The implementation provides optivis, which is the main framework for providing the visualisations of parameters learnt by the different layers of the deep neural network. It can be used to visualise torchvision models with no overhead for setup. Activation atlas methods, feature visualisa-

tion methods, building blocks, and differentiable image parameterisations can be used to visualise the different features learnt by the network. Activation atlas methods comprise different methods that show the network activations for a particular class or average activations in a grid cell of the image. Feature visualisation methods help understand the crucial features for a neuron, entire channel, or layer. Building blocks help to visualise the activation vector and its components for the given image. Differentiable image parameterisations find the types of image generation processes that can be backpropagated through, and this helps to perform appropriate preconditioning of the input image to improve the optimisation of the neural network.

2.4. The Pipeline: TorchEsegeta

The pipeline architecture is shown in Figure 1. The pipeline is implemented keeping in line with the object-oriented methodology. One of the key features of the pipeline is that it is easily scalable, according to the customised needs of the end-user because of the JSON-based configuration. The pipeline is a plug and play software and can be easily used by the end-users. This pipeline is publicly available on GitHub (TorchEsegeta on GitHub: https://github.com/soumickmj/TorchEsegeta).
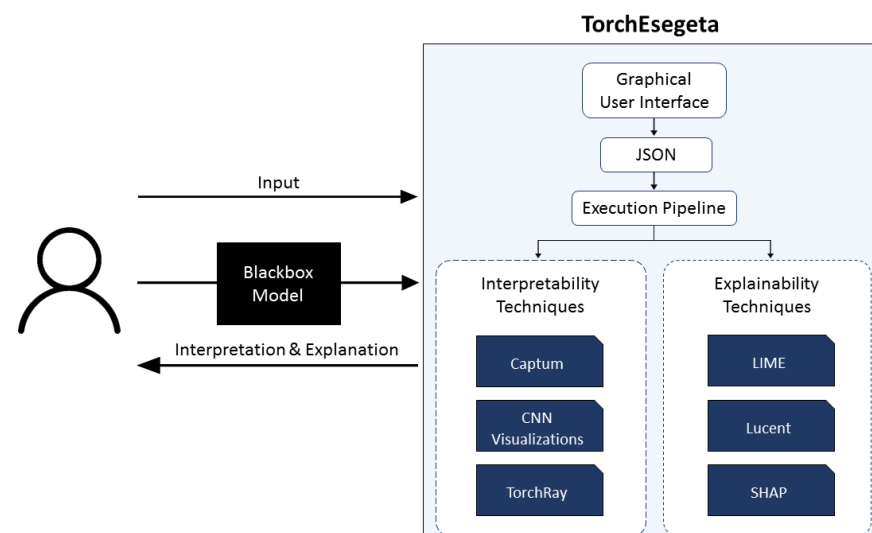

Figure 1. TorchEsegeta pipeline architecture.

2.5. Features of TorchEsegeta

Three-dimensional (3D) input data are especially important for medical images such as MRI and CT scans. Hence, the framework has been built to support both 2D and 3D input data. One of the most important features of the pipeline is scalability. New methods can be added to the pipeline seamlessly by the users and used as per requirement.

The JSON-based configuration will allow the user to execute the pipeline according to their specific needs and is easy to use the feature. The logging facility allows the users to track the pipeline execution and make changes in case of any error because of wrong parameter usage.

The timeout facility makes sure that no method execution is above a certain threshold of time. That way, it saves both computation resources as well as valuable time for the end-user.

2.5.1. Parallel Execution

Another way of saving much valuable time for the user is by using Multi-GPU, which helps in the parallel execution of multiple methods simultaneously. The multi-threading feature enables the execution of multiple methods on the same GPU, allowing the users to make optimum use of resources.

2.5.2. Patch-Based Execution

Patch-based models are supported in the pipeline and thus help with running computationally intensive methods. Existing interpretability and explainability methods do not give an option to be directly used on patch-based models.

2.5.3. Automatic Mixed Precision

Automatic Mixed Precision [36] facility is also incorporated in the framework, aiding in the faster execution of the code. Due to the use of this technique, the overall memory consumption while executing the methods is greatly reduced. The execution time is also reduced as a result.

2.5.4. Wrapper for Segmentation Models

The interpretability techniques that are available publicly are basically for classification problems. The framework is an extension to the segmentation problem with the help of a task-specific wrapper functionality—

1. Pixel-wise multi-class classification;
2. Threshold-based pixel classification.

Pixel-Wise Multi-Class Classification

In this method, the pixel scores are summed up for all pixels predicted as each class. Two main steps are performed:

a. Pixel-wise Class Assignment—In this step, for every pixel, the argmax class is computed. Let y be the image with dimensions

$$y_{ij} = \arg\max_k (y_{ijk}). \tag{15}$$

b. Final Output Calculation—In this step, the sum of the pixel scores for all pixels is predicted as each class is computed.

$$Out_{in} = count\{y_{ij} \mid y_{ij} \in class\ m\} \tag{16}$$

where $0 \leq m \leq N_c$.

Threshold-Based Pixel Classification

This method performs class identification by the Otsu threshold and then sums up the pixels for each class. This task is also performed in two steps:

a. Normalisation—In this step, the input image is normalised by the following function:

$$y_{ij_{norm}} = \frac{y_{ij} - min(y)}{max(y) - min(y)}. \tag{17}$$

b. Pixel-wise binarisation—The pixel-wise binarisation is performed with the help of Otsu thresholding.

$$y_{ij} = \begin{cases} 1 & where\ \ y_{ij_{norm}} > th \\ \\ 0 & elsewhere \end{cases} \tag{18}$$

where th = otsu($y_{ij_{norm}}$).

The output for both the processes is a tensor with a single value for each class.

2.5.5. Graphical User Interface

A Graphical User Interface (GUI) has been created to provide the end-user with an intuitive graphical layout to select the parameters and interpretability methods according to their requirement. The first screen shown in Figure 2 provides the user with a dialogue

box for parameter selection. The user can choose the model nature, model name, dataset, and many more run-time parameters. Once these selections are made, the 'Select Methods' button will display the following dialogue box, as shown in Figure 3. The users have a wide range of interpretability methods to choose from in this dialogue box. Once the interpretability methods are chosen, the method-specific parameter dialogue box is displayed to the user, as shown in Figure 4. Then, the users can choose the visualisation method, device ID, and many other parameters. On clicking the 'Next' button, the code will be executed in the back-end, and the output will be generated in the output path specified in Figure 2.
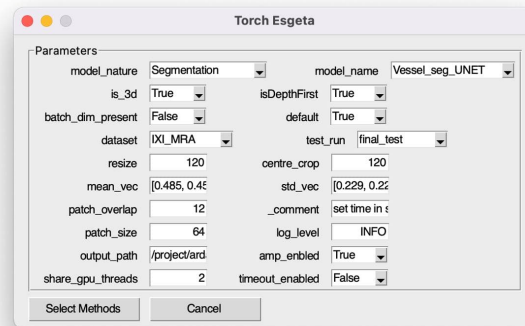

Figure 2. GUI: General parameter selection. Select the desired common parameters for running the pipeline, including the selection of the model.
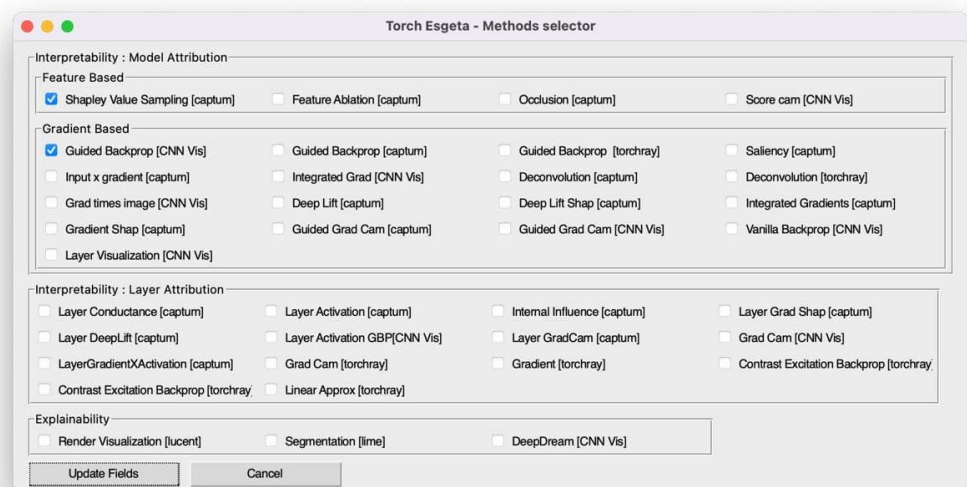

Figure 3. GUI: Method selection. Choosing which methods are to be applied on given models.
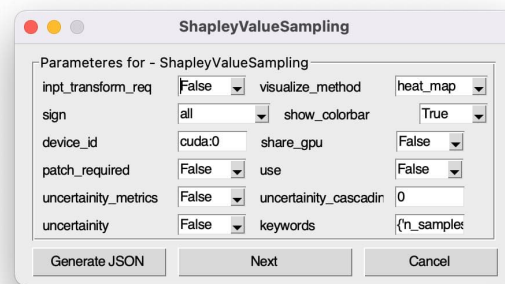
Figure 4. GUI: Method-specific parameter selection for a chosen method.

2.6. Evaluation Methods

In order to evaluate and compare the methods, both qualitative and quantitative aspects have been considered.

2.6.1. Qualitative Evaluation

For qualitative or visual evaluation, the cascading randomisation [37] technique was implemented, in which the model weights are randomised successively, from the top to the bottom layers. The learned weights of the layers are destroyed from top to bottom by this technique. While doing so, the interpretability and explainability techniques are applied to each state of randomisation. If the interpretability–explainability results are dependent upon the model's weights—how it is supposed to be—then the quality would decrease as the amount of randomisation increases. If they are not dependent, then the results will be unaffected by the randomisation—hence, the technique is not accurate or unsuitable for the current model.

2.6.2. Quantitative Evaluation

For the quantitative evaluation, the uncertainty method was chosen due to the unavailability of ground truth data for the attribution images. Two metrics have been used for computing the uncertainty values of the attribution methods: Infidelity [38]] and Sensitivity [38]. These metrics can be used only on model attribution methods as of now. It is to be noted that the qualitative metrics can be used on top of cascading randomisation as an additional level of evaluation.

Infidelity (https://captum.ai/api/metrics.html) represents the expected mean-squared error between the explanation multiplied by a meaningful input perturbation and the differences between the predictor function at its input and perturbed input. Sensitivity measures the extent of explanation change when the input is slightly perturbed.

3. Results

To evaluate the TorchEsegeta pipeline for segmentation models, a use-case of vessel segmentation was chosen.

3.1. Models

The segmentation network models chosen for this use case are from the DS6 paper [39]. The models are U-Net, U-Net MSS, and U-Net MSS with Deformation. The difference between the U-Net MSS and U-Net MSS with Deformation is a change in the number of upsampling and downsampling layers and a modified activation function. U-Net MSS performs downsampling five times using convolution striding and uses transported convolution for upsampling, and it includes instance normalisation and Leaky ReLU in the convolution blocks. On the contrary, the modified version applies four downsampling

layers using max-pool and performs upsampling using interpolation combined with Batch normalisation and ReLU in its convolution block.

For U-Net MSS with Deformation, a small amount of variable elastic deformation is added at the time of training, along with each volume input. The authors have given a comprehensive overview of the method, and based on their results, one can see that U-Net MSS with Deformation is the best-performing model.

3.2. Use Case Experiment

As a use case study, the previously mentioned interpretability and explainability techniques were explored while analysing the results of a vessel segmentation model trained on Time-of-Flight (TOF) Magnetic Resonance Angiogram (MRA) images of the human brain called DS6 [39]. The model automatically segments vessels from 3D 7 Tesla TOF-MRA images. A study about the attribution outputs of the different methods across the three models was conducted. The zoomed-in portions of the attribution images show even more closely the model's focus areas. Figure 5 shows examples of interpretability techniques compared across the three models. Figure 6 portrays similar interpretability results from the U-Net MSS Deformation model. By observing the interpretability results, it can be understood where the network is focusing. In production, ground-truth segmentations are not available. By looking at the interpretability results, it might be possible to estimate in which regions the network might mispredict by looking at the areas where the network did not adequately focus. These interpretability results can assist radiologists to build trust in models that focus on the correct regions. On the other hand, a model developer can benefit from looking at the wrong focus regions or regions with no focus, because it is knowledge that can be further used to improve the model architecture or the training process.
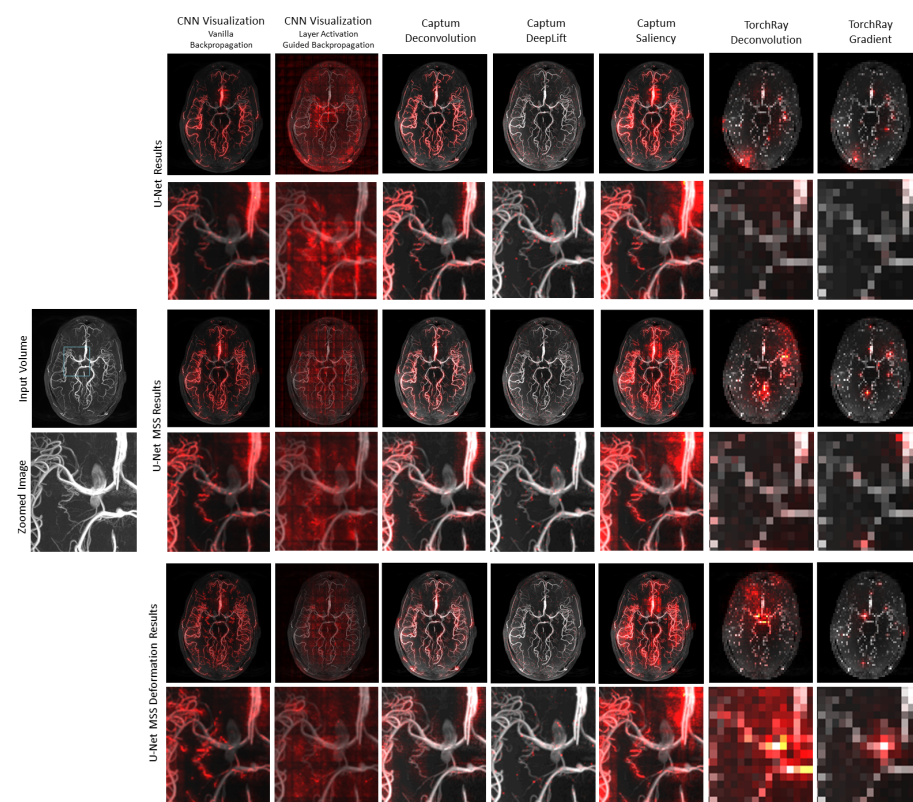

Figure 5. Example of interpretability techniques compared across three models: U-Net, U-Net MSS, and U-Net MSS with deformation, along with its corresponding zoomed image. The higher the intensity of red, the higher the concentration of the focus of the network. Looking at the regions where the network did not focus, it can be understood which parts of the segmentation prediction might be wrong.
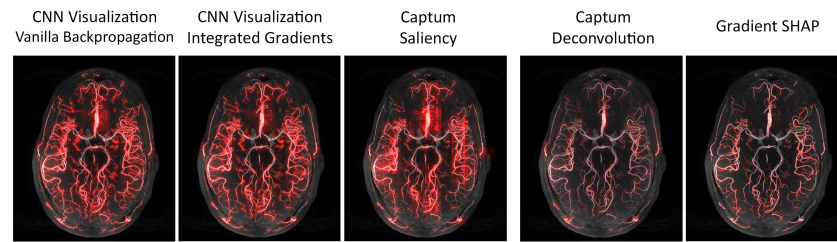
Figure 6. Some of the similar interpretability results of the U-Net MSS Deformation model. The higher the intensity of red, the higher the concentration of the focus of the network. The results of CNN Visualization Vanilla Backpropagation, CNN Visualization Integrated Gradients, and Captum Saliency show similar focus areas: on the anterior, posterior, right, and left regions of the brain. Captum Deconvolution and Gradient SHAP emphasise specifically on the cerebral artery itself, not the whole area.

In all of these attribution maps, whichever pixel is highlighted by red represents high activation, indicating the most important pixels in the input according to the respective method, and the other region represents lower to no activation.

3.3. Notable Observations

In Figure 5, the CNN Visualization Layer Activation Guided Backpropagation attribution map shows a lot of attributions, most of which are on parts other than the brain vessels, i.e., the region of interest. The respective zoomed images also confirm the observation. On the contrary, methods such as Captum Deeplift and TorchRay Gradient provide fewer attributions and miss out on major portions of the brain vessels. Torchray Deconvolution provides more attributions comparatively; however, most of the attributions are concentrated on certain areas of the brain. CNN Visualisation Vanilla Backpropagation, Captum Deconvolution, and Captum Saliency provide better results compared to the others—they provided more human-interpretable results. These methods mainly attribute on the region of interest. However, a look at the respective zoomed-in images would reveal that for Captum Saliency, there are attributions even in the areas surrounding the brain vessels. For the other two, the attributions are mainly on the brain vessels only. Figure 6 shows a comparison of some of the better performing interpretability methods for the U-Net MSS Deformation model.

In Figures 7 and 8, the layer-wise attributions for the three models are shown for the following methods: Excitation Backpropagation, Layer Conductance, Layer DeepLift, and Layer Gradient X Activation. For all the methods, the maximum attributions are shown in the Conv3 layer. Among the three models, U-Net MSS Deformation seems to focus better than the other two, as it attributions can be seen all over the brain contrary to the other two (it is to be noted the vessels are present all over the brain and not focused in a specific region). The network focus shifts while going from the first to the last layer of the model. In the initial layers, the focus is more on selective regions of the brain. However, towards the deeper layers such as Conv3, the focus is almost on the entire brain.

3.4. Evaluation

A comparative study of the cascading randomisation technique is shown for the outputs of four interpretability methods for all the models in Figure 9.

Moreover, to show the functionality of the quantitative evaluation part of the pipeline, a few methods were compared quantitatively using infidelity and sensitivity; the scores are shown in Tables 1 and 2.
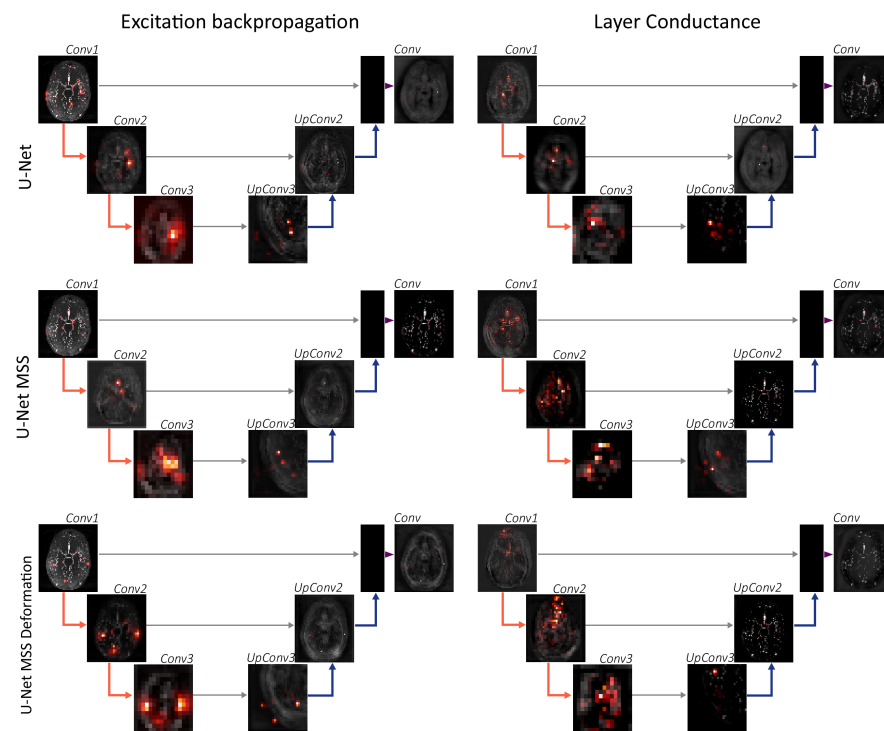
Figure 7. Layer-based interpretability methods: Excitation Backpropagation and Layer Conductance, shown using representative U-Nets. The higher the intensity of red, the higher the concentration of the focus of the network. This figure shows how the focus of the network changes in each layer for the three different models.

Figure 8. Layer-based interpretability methods: Layer Deeplift and Layer Gradient X Activation, shown using representative U-Nets. The higher the intensity of red, the higher the concentration of the focus of the network. This figure shows how the focus of the network changes in each layer for the three different models.
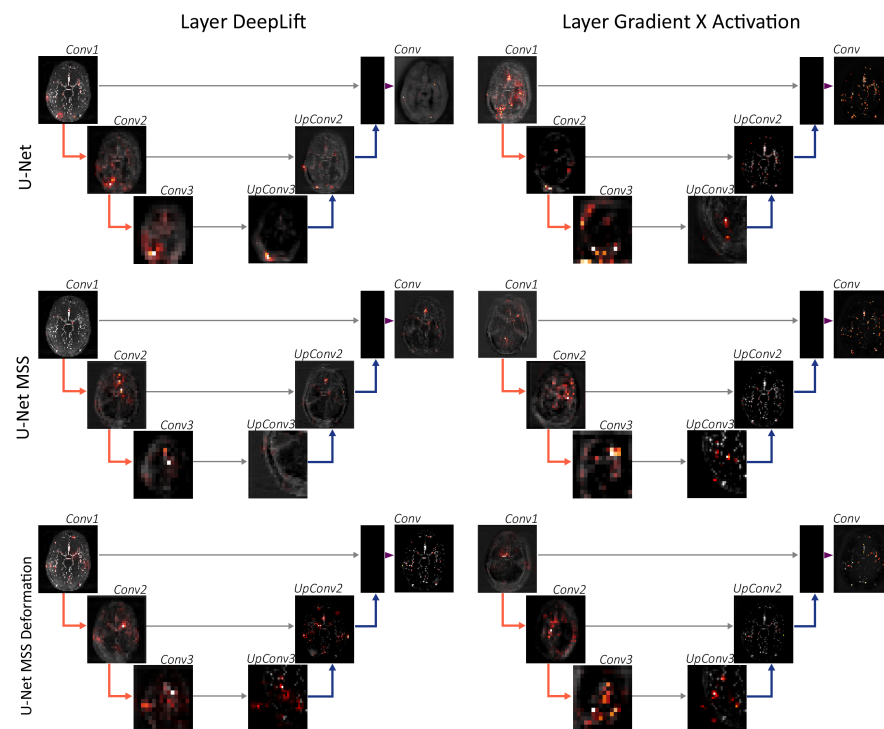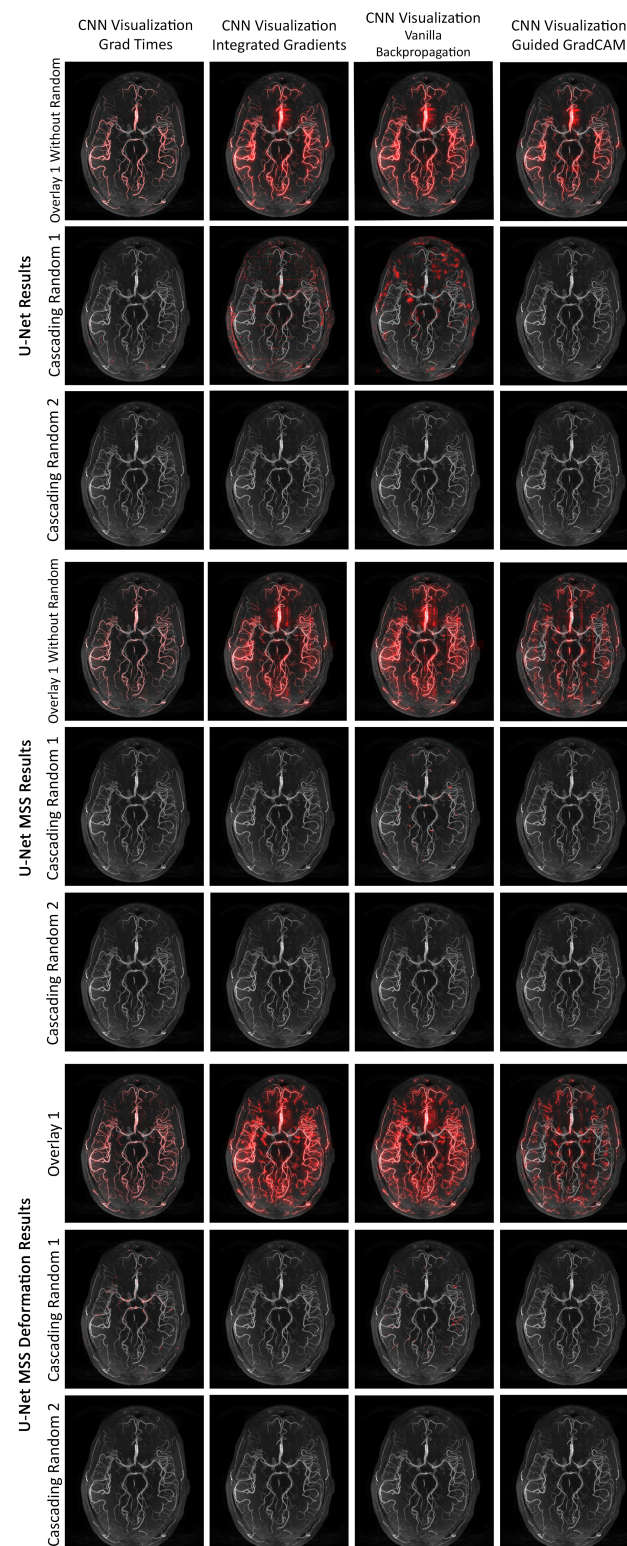
Figure 9. Example of cascading randomisation outputs of the three models: U-Net, U-Net MSS, and U-Net MSS with deformation. The outputs of all models and methods show the focus without randomisation fading through cascading randomisation 1 and 2. This implies that the output of these interpretability methods relies upon the weights of the network and not just random predictions. CNN Visualisation-Guided GradCAM shows the strongest dependency on the weights as even with cascading 1, all the attributions disappeared. Comparing the attributions for U-Net against the other two models, it can be observed that the dependency on weights is stronger for the other models.

Table 1. Infidelity scores.

	U-Net	U-NetMSS	U-NetMSS_Deform
Guided Backprop	5.87e−17	2.08e−17	2.59e−18
Deconvolution	3.20e−16	2.62e−16	1.48e−16
Saliency	1.95e−15	1.54e−15	1.23e−16

Table 2. Sensitivity Scores.

	U-Net	U-NetMSS	U-NetMSS_Deform
Guided Backprop	1.156	0.917	0.831
Deconvolution	1.210	1.188	1.140
Saliency	1.171	1.197	1.153

4. Discussion

In this work, various interpretability and explainability methods were adopted for segmentation models and were used to interpret the network. A pipeline has been developed and made public on GitHub (TorchEsegeta on GitHub: https://github.com/soumickmj/TorchEsegeta), TorchEsegeta, comprised of those interpretability and explainability methods that can be applied on 2D or 3D image-based deep learning models for classification and segmentation. The pipeline was experimented with using three models—U-Net, U-NetMSS and the U-NetMSS-Deform—for the task of vessel segmentation from MRAs, using interpretability methods. The evaluation of the methods has been done qualitatively using cascading randomisation and quantitatively using evaluation metrics. It is worth mentioning that several explainability methods are part of the pipeline, but they were not evaluated with any use-case scenario during this research.

This pipeline can be used by data scientists to improve their models by tweaking their models based on the shown interpretability and explainability to improve the reasoning of the models, in turn improving the performance. Moreover, they can use this pipeline to show it to the model's users, for them to have trust in the model while using them in high-risk situations. On the other hand, this pipeline can be used by expert decision-makers, such as clinicians, as a decision support system; by understanding the reasoning done by the models, they can get assistance in decision making.

It is noteworthy that the current pipeline can be extended for reconstruction purposes, and new interpretability and explainability methods can be added to the existing pipeline. Ground truth-based pixel-wise interpretability for segmentation models can be implemented, which will add a new dimension to the existing work. Nevertheless, the real evaluation of the interpretations and explanations can only be done by the domain experts—in this case of vessel segmentation, by the clinicians—who can judge whether these results are actually useful or not. This step of the evaluation was not performed under the scope of the current work and will be performed in the near future.

Author Contributions: Conceptualisation, S.C.; Literature Survey, A.S. and R.N.R.; Architecture Design, S.C. and A.D.; Pipeline Development, A.D., C.M. and B.M.; Experiments, A.D.; Quantitative Evaluation, M.V.; GUI Development, B.M.; Visualisation, C.S.; Writing—original draft, S.C., C.M. and R.N.R.; Writing—review and editing, S.C., O.S. and A.N. All authors have read and agreed to the submitted version of the manuscript.

Data Availability Statement: Data used for the use case experiment in this paper is not publicly available due to data privacy and security concerns regarding medical data. The manuscript of the dataset is: "Mattern et al. 2018. Prospective motion correction enables highest resolution time-of-flight

angiography at 7T. Magnetic Resonance in Medicine 80, 248–258. doi:10.1002/mrm.27033". Data might be available on request from the corresponding author of that original manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Marcinkevičs, R.; Vogt, J.E. Interpretability and explainability: A machine learning zoo mini-tour. *arXiv* **2020**, arXiv:2012.01805.
2. Chakraborty, S.; Tomsett, R.; Raghavendra, R.; Harborne, D.; Alzantot, M.; Cerutti, F.; Srivastava, M.; Preece, A.; Julier, S.; Rao, R.M.; et al. Interpretability of deep learning models: A survey of results. In Proceedings of the 2017 IEEE Smartworld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI), San Francisco, CA, USA, 4–8 August 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
3. Emmert-Streib, F.; Yli-Harja, O.; Dehmer, M. Explainable artificial intelligence and machine learning: A reality rooted perspective. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2020**, *10*, e1368. [CrossRef]
4. Belle, V.; Papantonis, I. Principles and practice of explainable machine learning. *arXiv* **2020**, arXiv:2009.11698.
5. Dubost, F.; Bortsova, G.; Adams, H.; Ikram, A.; Niessen, W.J.; Vernooij, M.; De Bruijne, M. Gp-unet: Lesion detection from weak labels with a 3d regression network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 214–221.
6. Gu, R.; Wang, G.; Song, T.; Huang, R.; Aertsen, M.; Deprest, J.; Ourselin, S.; Vercauteren, T.; Zhang, S. CA-Net: Comprehensive attention convolutional neural networks for explainable medical image segmentation. *IEEE Trans. Med. Imaging* **2020**, *40*, 699–711. [CrossRef]
7. Arrieta, A.B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* **2020**, *58*, 82–115. [CrossRef]
8. Choo, J.; Liu, S. Visual analytics for explainable deep learning. *IEEE Comput. Graph. Appl.* **2018**, *38*, 84–92. [CrossRef] [PubMed]
9. Fong, R.; Patrick, M.; Vedaldi, A. Understanding deep networks via extremal perturbations and smooth masks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019 ; pp. 2950–2958.
10. Chatterjee, S.; Saad, F.; Sarasaen, C.; Ghosh, S.; Khatun, R.; Radeva, P.; Rose, G.; Stober, S.; Speck, O.; Nürnberger, A. Exploration of interpretability techniques for deep covid-19 classification using chest x-ray images. *arXiv* **2020**, arXiv:2006.02570.
11. Ozbulak, U. PyTorch CNN Visualizations. 2019. Available online: https://github.com/utkuozbulak/pytorch-cnn-visualizations (accessed on 10 July 2021).
12. Samek, W.; Montavon, G.; Lapuschkin, S.; Anders, C.J.; Müller, K.R. Toward interpretable machine learning: Transparent deep neural networks and beyond. *arXiv* **2020**, arXiv:2003.07631.
13. Lundberg, S.; Lee, S.I. A unified approach to interpreting model predictions. *arXiv* **2017**, arXiv:1705.07874.
14. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
15. Fisher, A.; Rudin, C.; Dominici, F. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *J. Mach. Learn. Res.* **2019**, *20*, 1–81.
16. Castro, J.; Gómez, D.; Tejada, J. Polynomial calculation of the Shapley value based on sampling. *Comput. Oper. Res.* **2009**, *36*, 1726–1730. [CrossRef]
17. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 818–833.
18. Petsiuk, V.; Das, A.; Saenko, K. RISE: Randomized Input Sampling for Explanation of Black-Box Models. *arXiv* **2018**, arXiv:1806.07421
19. Wang, H.; Wang, Z.; Du, M.; Yang, F.; Zhang, Z.; Ding, S.; Mardziel, P.; Hu, X. Score-CAM: Score-weighted visual explanations for convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 24–25.
20. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv* **2013**, arXiv:1312.6034.
21. Springenberg, J.T.; Dosovitskiy, A.; Brox, T.; Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv* **2014**, arXiv:1412.6806.
22. Shrikumar, A.; Greenside, P.; Shcherbina, A.; Kundaje, A. Not just a black box: Learning important features through propagating activation differences. *arXiv* **2016**, arXiv:1605.01713.
23. Sundararajan, M.; Taly, A.; Yan, Q. Axiomatic attribution for deep networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017 ; pp. 3319–3328.
24. Shrikumar, A.; Greenside, P.; Kundaje, A. Learning important features through propagating activation differences. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 3145–3153.
25. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 618–626.

26. Chattopadhay, A.; Sarkar, A.; Howlader, P.; Balasubramanian, V.N. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 839–847.

27. Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; Wattenberg, M. Smoothgrad: Removing noise by adding noise. *arXiv* **2017**, arXiv:1706.03825.

28. Mahendran, A.; Vedaldi, A. Understanding Deep Image Representations by Inverting Them. *arXiv* **2014**, arXiv:c1412.0035

29. Dhamdhere, K.; Sundararajan, M.; Yan, Q. How important is a neuron? *arXiv* **2018**, arXiv:1805.12233.

30. Leino, K.; Sen, S.; Datta, A.; Fredrikson, M.; Li, L. Influence-directed explanations for deep convolutional networks. In Proceedings of the 2018 IEEE International Test Conference (ITC), Phoenix, AZ, USA, 29 October–1 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–8.

31. Zhang, J.; Bargal, S.A.; Lin, Z.; Brandt, J.; Shen, X.; Sclaroff, S. Top-down neural attention by excitation backprop. *Int. J. Comput. Vis.* **2018**, *126*, 1084–1102. [CrossRef]

32. Liu, H.; Brock, A.; Simonyan, K.; Le, Q.V. Evolving normalization-activation layers. *arXiv* **2020**, arXiv:2004.02967.

33. Guo, M.; Zhang, Q.; Liao, X.; Zeng, D.D. An Interpretable Neural Network Model through Piecewise Linear Approximation. *arXiv* **2020**, arXiv:2001.07119

34. Ancona, M.; Ceolini, E.; Öztireli, C.; Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv* **2017**, arXiv:1711.06104.

35. 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, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144.

36. Micikevicius, P.; Narang, S.; Alben, J.; Diamos, G.; Elsen, E.; Garcia, D.; Ginsburg, B.; Houston, M.; Kuchaiev, O.; Venkatesh, G.; et al. Mixed precision training. *arXiv* **2017**, arXiv:1710.03740.

37. Adebayo, J.; Gilmer, J.; Muelly, M.; Goodfellow, I.; Hardt, M.; Kim, B. Sanity checks for saliency maps. *arXiv* **2018**, arXiv:1810.03292.

38. Yeh, C.K.; Hsieh, C.Y.; Suggala, A.S.; Inouye, D.I.; Ravikumar, P. On the (in) fidelity and sensitivity for explanations. *arXiv* **2019**, arXiv:1901.09392.

39. Chatterjee, S.; Prabhu, K.; Pattadkal, M.; Bortsova, G.; Sarasaen, C.; Dubost, F.; Mattern, H.; de Bruijne, M.; Speck, O.; Nürnberger, A. DS6, Deformation-aware Semi-supervised Learning: Application to Small Vessel Segmentation with Noisy Training Data. *arXiv* **2020**, arXiv:2006.10802.