Combining an Autoencoder and a Variational Autoencoder for Explaining the Machine Learning Model Predictions

Lev Utkin, Pavel Drobintsev, Maxim Kovalev, Andrei Konstantinov
Peter the Great St.Petersburg Polytechnic University
St.Petersburg, Russia
lev.utkin@gmail.com, drob@ics2.ecd.spbstu.ru, maxkovalev03@gmail.com, andrue.konst@gmail.com

***Abstract*—A method for explaining a deep learning model prediction is proposed. It uses a combination of the standard autoencoder and the variational autoencoder. The standard autoencoder is exploited to reconstruct original images and to produce hidden representation vectors. The variational autoencoder is trained to transform the deep learning model outputs (embedding vectors) into the hidden representation vectors of the standard autoencoder. In explaining or testing phase, the variational autoencoder produces a set of vectors based on the explained image embedding. Then the trained decoder part of the standard autoencoder reconstructs a set of images which form a heatmap explaining the original explained image. In fact, the variational autoencoder plays a role of the perturbation technique of images. Numerical experiments with the well-known datasets MNIST and CIFAR10 illustrate the propose method.**

I. Introduction

In spite of successful and widespread application of the deep learning techniques, the deep learning models are regarded as black-boxes, and they suffer from a lack of explainability, that is, they do not explain their predictions in a way that humans could understand. It turns out that a model cannot be used in many applications when its predictions do not have the corresponding explanation or interpretation, when the predictions cannot be understood, when it is impossible to answer the question why the deep learning model provides some prediction. For example, a doctor has to have an explanation of a diagnosis stated by the machine learning model in order to choose a corresponding treatment [1]. Therefore, explainers or special meta-models explaining and interpreting the classification and regression model predictions have been developed to provide more confidence and acceptability for the deep learning models used in practice [2], [3], [4], [5].

In fact, the explanation model allows us to determine what features of an example or a set of examples impact on the black-box model prediction. It should be pointed out that we consider the black-box model which means that we do not know or do not use any its details. Moreover, we consider post-hoc explanations which involve auxiliary models to explain the black-box models after it has been trained.

Among variety of explanation models, we select local models which try to explain what features lead to the individual prediction [5]. Explanations are derived by fitting an interpretable model locally around the considered example. In contrast to local explanation models, the global ones try to explain the black-box model on the whole dataset or its part.

One of the important techniques applied to explanation is the perturbation technique [6], [7], [8]. It assumes that contribution of a feature can be determined by measuring how prediction score changes when the feature is altered [9]. This technique can be applied to a black-box model without any need to access the internal structure of the model. It assumes that a feature is important and strongly impacts on the outcome if its change sufficiently changes the outcome. However, the perturbation technique may meet computational difficulties when the perturbed input examples have a lot of features. Moreover, perturbations may lead to change of the input data sense, for example, they may lead to transition of data from one class to another class. A closest target class sample can be just unrealistic. Some perturbation-based methods [10] measure the amount of change in prediction when each pixel is removed individually. However, many typical classifiers, for example, neural networks, cannot make a prediction for a partially removed input. It should be noted that there are deep learning models, for example, the Siamese neural networks [11], [12] for which direct distances between input feature vectors often do not have a sense because there is only similarity and dissimilarity relationships between input vectors. As a result, the perturbation technique cannot be used directly for inputs of deep learning models.

In order to overcome these difficulties, we propose an explanation method which uses a specific autoencoder (AE) as a part of the explainer to reconstruct the input examples. It also uses a variational autoencoder (VAE) [13] for generating new embeddings. In fact, the VAE plays a role of the perturbation technique of images. It helps confine the sample in the realistic sample space. However, the VAE in our method does not generate new images, it generates new embeddings for the AE which uses them for reconstructing an important feature heatmap. This is a very important and distinctive feature of the approach.

An interesting example of applying the proposed approach is the Siamese neural network (SNN) explanation. The SNN consists of two identical neural subnets sharing the same set of weights. The basic idea behind the SNN is to train the subnets to compare a pair of feature vectors in terms of their semantic similarity or dissimilarity. The SNN realizes a non-linear embedding of data with the objective to bring together similar instances and to move apart dissimilar instances.

The proposed method is very efficient, and it is illus-

trated by numerical examples with the datasets MNIST and CIFAR10.

The paper is organized as follows. Related work is in Section II. Basic concepts of the AE and VAE can be found in Section III. Basic ideas behind the proposed explanation algorithm are considered in Section IV. The corresponding application of the algorithm to explaining SNNs is illustrated in Section V. Numerical experiments with real data are given in Section VI. Concluding remarks can be found in Section VII.

II. Related work

Local explanation methods. The explanation methods can be regarded as an important supplement of deep learning models in many applications. Hence, a lot of methods have been proposed to explain black-box models locally. One of the most popular local explanation methods is the Local Interpretable Model-agnostic Explanations (LIME) [14], which is based on using linear models to approximate the predictions of black-box models locally. The linear approximation of some unknown function of the input features produced by the black-box model in the local area around the example of interest explains by considering coefficients of the features as quantitative impacts on the prediction. Following the original LIME [14], a lot of its modifications have been developed due to a simple idea behind the method to construct a linear approximating model, for example, ALIME [15], NormLIME [16], Anchor LIME [17], GraphLIME [18], LIME for text data [19], SurvLIME [20]. Another explanation method, which is also based on the linear approximation, is the SHAP [21], [22], which is based on a game-theoretic approach and on using Shapley values. A large part of methods uses the so-called counterfactual explanations [23] which answer the question why outcome Y is provided by the black-box model instead of Z.

An important group of explanation methods is based on perturbation techniques [6], [7], [24], [8], which are also used in LIME. A large part of explanation methods uses a prototype technique which selects representative instances from training data, for instance, from instances belonging to the same class. These instances are called prototypes [25], [26]. The main idea behind the explanation methods based on this technique is to find out how an explained example is similar to a prototype.

A lot of explanation methods, their analysis, their critical review can be found in survey papers [27], [28], [29], [30], [3], [31], [32]. A comprehensive analysis of the explanation method pitfalls and incorrect assumptions accepted in the explanation models is provided by Rudin [31].

Visual explanation methods. Among the available explanation methods, we select the methods of visual explanation [33], [34], [35], [36], [37], [38], [39], [40]. They mainly highlight "important" regions or subsets of features in the input image, which lead the model to make its prediction. These methods are very important for explaining CT images.

Petsiuk et al. [24] proposed to generate a set of random masks and to use the predicted class probabilities as weights, computing a weighted sum of the masks as the saliency map. An extension of this method for generating visual explanations was proposed in [41]. An approach for saliency map generation for black-box models using a Bayesian optimization sampling method is provided by [42]. Goyal et al. [37] considered counterfactual visual explanations and studied how such counterfactual visual explanations can be generated to explain the decisions of deep vision systems by identifying what and how regions of an input image would need to change in order for the system to produce a specified output. Directed perturbations in the examples to observe which attribute values change when classifying the examples into the counter classes was introduced by Gulshad and Smeulders [43].

Autoencoders and variational autoencoders for explanation. It should distinguish explanations of the AE results and explanations by means of AEs. The first direction was studied by Anwarg et al. [44] where the authors proposed a method for explaining anomalies detected by the AE. A lot of works are devoted to the second direction. In particular, Bellini et al. [45] exploited a semantics-aware AE to compute explainable recommendations. Guidotti et al. [46] presented an explanation method which exploits the latent feature space learned through an adversarial autoencoder [47]. Shankaranarayana and Runje [15] proposed modifications of LIME by employing an AE, which serves as a better weighting function for the local model. Qi and Li [48] proposed Sparse Reconstruction Autoencoder (SRAE) for learning the embedding to the explanation space. SRAE aims to reconstruct part of the original feature space while retaining faithfulness. Haghighi et al. [49] considered an explainable recommendation system using an autoencoder model whose predictions can be explained using the neighborhood based explanation style. Gee et al. [50] adopted an autoencoder-prototype architecture for explaining the deep classification of time-series data.

VAEs have been also applied to solving the explanation problem. Schockaert et al. [51] proposed the method VAE-LIME for local interpretability of data-driven models forecasting the temperature of the hot metal produced by a blast furnace. The VAE in [51] aims to generate optimal synthetic examples to train a local interpretable model. Uzunova et al. [52] applied the VAE to generating equivalent deletions of pathologies to enhance the reliability of diagnosis explanations in the brain lesion MRI investigation. Alvarez-Melis and Jaakkola [53] provided explanations of the text classification results in the form of partitioned graphs by using the VAE.

III. Autoencoder and variational autoencoder

An AE is a a feed-forward neural network that is trained to learn reconstructions that are close to its original input. The AE allows us to get a low-dimensional and non-linear feature representation of the input data. In other words, the conventional AE learns a mapping from high-dimensional inputs to the low-dimensional encoded representation which can be used to reconstruct the original input. It should be noted that the VAE is also used to learn a lower-dimensional feature representation from the unlabeled training data. However, in contrast to the conventional AE, the VAE is a generative model. It does not replicate the same input as the AE does, but randomly samples from the latent space or generates variations on the input from a continuous latent space. Therefore, one of the main aims of the VAE is to generate new data related to the original source data.

Let $S = \{\mathbf{x}_i, i = 1, ..., n\}$ be a dataset consisting of feature vectors $\mathbf{x}_i \in \mathbb{R}^m$. AEs consist of two parts. The first one, encoder, provides a mapping from an input domain, $\mathcal{X}$, to a code or embedding domain, $\mathcal{C}$, i.e., the hidden representation. The second part, decoder, maps from $\mathcal{C}$ back to $\mathcal{X}$. Let us denote the encoding function as $f_E(\mathbf{x}, W_E)$ and the decoding function as $f_D(\mathbf{h}, W_D)$. Here $\mathbf{x} \in \mathbb{R}^m$ is the input vector of size m, $\mathbf{h} \in \mathbb{R}^D$ is the embedding vector of size D such that $\mathbf{h} = f_E(\mathbf{x}, W_E)$; W_E and W_D are matrices of weights of the encoder and the decoder, respectively. Then the reconstruction loss function for learning the AE is defined as

$$L_{\text{rec\_AE}}(W_E, W_D) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}_i^*\|_2^2 + \lambda R(W).$$

Here $\mathbf{x}_i^*$ is the reconstructed vector such that $\mathbf{x}_i^* = f_D(\mathbf{h}_i, W_D)$; $R(W_E, W_D)$ is a regularization term added to improve generalization of the AE;
$W = W_E \cup W_D$; λ is a hyper-parameter which controls the strength of the regularization.

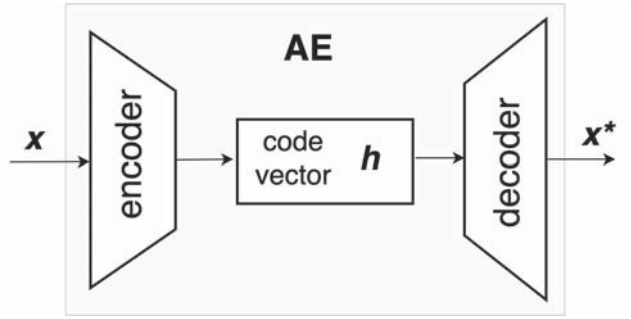

Fig. 1. A scheme of the AE

The main difference between a VAE and an autoencoder is that the VAE is a stochastic generative model that can give calibrated probabilities, while an autoencoder is a deterministic discriminative model that does not have a probabilistic foundation [54]. The VAE also consists of two parts: a generative part (the coder) and an inference part (the decoder). In the generative part, a probabilistic decoder reproduces $\mathbf{x}^*$ close to an observation $\mathbf{x}$ from a latent variable $\mathbf{z} \sim p(\mathbf{z})$, i.e., $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}) = p_\theta(\mathbf{x}|\zeta)$, where ζ is obtained from a latent variable $\mathbf{z}$ by the neural network which realizes a function $\zeta = f_D(\mathbf{z})$. In the inference part, a probabilistic encoder outputs a latent variable $\mathbf{z} \sim p_\varphi(\mathbf{z}|\mathbf{x}) = p_\varphi(\mathbf{z}|\nu)$, where ν is computed from the observation $\mathbf{x}$ by the corresponding neural network which realizes a function $\nu = f_E(\mathbf{x})$. The latent variable $\mathbf{z}$ is generated from $p_\varphi(\mathbf{z}|\mathbf{x})$, and the output vector $\mathbf{x}^*$ is generated from $p_\theta(\mathbf{x}|\mathbf{z})$. The model parameters θ and φ are jointly learned with the stochastic gradient method through the backpropagation. In the original VAE [13], the prior distribution of $p(\mathbf{z})$ is assumed to be normally distributed. A general scheme of the VAE is shown in Fig. 2. The encoder output is a set of pairs (m_i, s_i), where m_i and s_i are the expectation and standard deviation of the normal distribution such that the vectors $\mathbf{z}$ are generated in accordance with this distribution. In sum, the VAE generates a set of output vectors $\mathbf{x}^*$ which are close to the input vector $\mathbf{x}$.

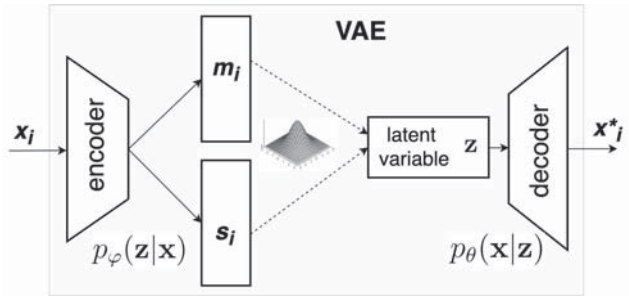

Fig. 2. A general scheme of the VAE

IV. The proposed algorithm for the black-box model explanation

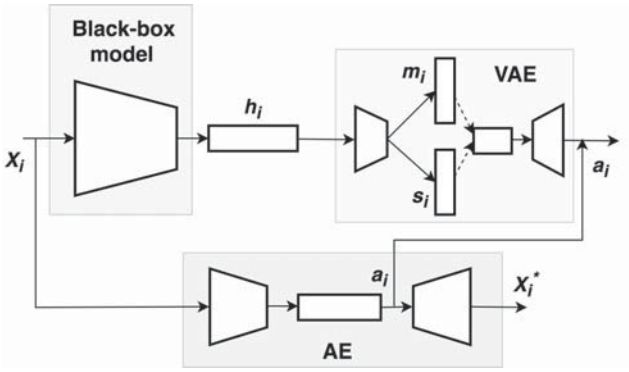

Fig. 3. A scheme of the algorithm for explaining the black-box model

The proposed algorithm for explaining the SNN consists of several steps. In order to avoid misunderstanding in description of different objects of interest, we will call original and reconstructed examples by images. Embeddings and other feature representations will be called by vectors.

The first step is to train an AE by using examples (images) $\mathbf{x}_1, ..., \mathbf{x}_n$ from the training set S, $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, ..., n$. This step has a double aim. First, we aim to get the code (the hidden representation) of the AE denoted as $\mathbf{a}_i \in \mathbb{R}^D$ for every $\mathbf{x}_i$. Its length D coincides with the length of the embedding $\mathbf{h}_i$ which is the outcome of the explained black-box model. This code will be used to train the VAE for generating new hidden representation vectors. Second, the decoder of the trained AE will allow us to get the reconstructed images $\mathbf{x}_i^*$ from the codes $\mathbf{a}_i$, which are close to the original images $\mathbf{x}_i$.

The main difference of the proposed AE from the standard one is its reconstruction loss function. The reconstruction loss function $L_{\text{rec\_AE}}$ for training of the AE is of the form:

$$L_{\text{rec\_AE}}(W) = \gamma \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}_i^*\|_2^2 + \mu \sum_{i=1}^{n} \|\mathbf{h}_i - \mathbf{a}_i\|_2^2 + \lambda R(W).$$

Here the first term is the standard reconstruction loss; the second term contributes into closeness of the hidden representation $\mathbf{a}_i$ and the black-box model output vector $\mathbf{h}_i$; $R(W)$ is a

regularization term; λ is a hyper-parameter which controls the strength of the regularization; γ and μ are weights that control the interaction of the loss function terms; $W = W_E \cup W_D$ is the set of the neural network weights. It should be noted that the second term of the loss function is very important because the trained decoder of the AE will be used for reconstruction of the explained image, and the trained encoder will play a role of a simplified representation of the black-box neural network. In other words, we would like to make the black-box model output $\mathbf{h}_i$ and the AE hidden representation $\mathbf{a}_i$ as close as possible to each other. The trade-off between closeness of $\mathbf{x}_i$ and $\mathbf{x}_i^*$ (the first term) and closeness of $\mathbf{h}_i$ and $\mathbf{a}_i$ is determined by the tuning parameters γ and μ.

The second step of the proposed algorithm is to train a generative neural network which is similar to the VAE. In fact, we have again the VAE, but in contrast to the original VAE, its output and input vectors are different. The input vector is $\mathbf{h}_i$, and the output vector is $\mathbf{a}_i$ which simultaneously is the hidden representation vector of the AE trained at the first step. The proposed VAE also has a double aim. First, we try to implement a function $g : \mathbb{R}^D \to \mathbb{R}^D$ between the black-box model output $\mathbf{h}_i$ and the AE output $\mathbf{a}_i$. The VAE plays a role of correction of vectors $\mathbf{h}_i$ to the hidden representation vectors of the AE. Second, according to the VAE function, we generate a set of vectors $\mathbf{a}_k$ which are close to some vector $\mathbf{a}$. This procedure can be implemented by means of the trained VAE. In fact, the generation is equivalent to perturbation of the vector $\mathbf{a}$, which is used in order to determine how changes of features of the embeddings impact on changes of features of reconstructed images. In other words, we implement the perturbation technique with this approach, which allows us to find the important features.

Before training the VAE and implementing the second step, we have two corresponding sets: a set of the AE hidden representations $\mathbf{a}_i \in \mathbb{R}^D$ for all original images $\mathbf{x}_i$, and a set of the black-box model outputs $\mathbf{h}_i \in \mathbb{R}^D$ (the embedding vectors) such that $\mathbf{h}_i = f(\mathbf{x}_i)$, $i = 1, ..., n$. Then a new dataset consisting of pairs $(\mathbf{h}_i, \mathbf{a}_i)$ can be constructed. This dataset is a basis for implementing the function g that maps every vector $\mathbf{h}_i$ obtained from the black-box model to the corresponding vector $\mathbf{a}_i$ or its close reconstruction. The obtained dataset is used to train the VAE of a special form, whose input and output are $\mathbf{h}_i$ and $\mathbf{a}_i$, respectively.

As a result of the above two steps, we have the trained AE, implementing functions $\mathbf{a}_i = f(\mathbf{x}_i)$ and $\mathbf{x}_i^* = f^*(\mathbf{a}_i)$, the trained VAE, implementing functions $\mathbf{a}_i = g(\mathbf{h}_i)$.

The third step of the proposed algorithm aims to perturb the reconstructed images in a special way in order to find their important features which explain the black-box model output. The main idea behind this step is to generate new embeddings corresponding to analyzed examples and reconstruct new images with respect to these embeddings. These reconstructed images indicate which features are changed. In fact, we do not perturb images or embeddings. The perturbation is implicitly realized by means of generating new embeddings $\mathbf{a}_k$ by the VAE.

In sum, the perturbed vectors $\mathbf{a}_k$ are fed to the AE decoder in order to get the reconstructed images $\mathbf{x}_k^*$ of the analyzed input example, which can be regarded as perturbed images. If we get N vectors $\mathbf{a}_k$, $k = 1, ..., N$, corresponding to the image $\mathbf{x}$, then the important features can be found by computing the following "residual" image:

$$\delta = N^{-1} \sum_{k=1}^{N} \left| \mathbf{x}_k^* - \mathbf{x}^{\#} \right|,$$

where $\mathbf{x}^{\#}$ is the mean image computed as $\mathbf{x}^{\#} = N^{-1} \sum_{k=1}^{N} \mathbf{x}_k^*$.

The image δ can be viewed as a heatmap explaining what features impact on the black-box model prediction or the model output. We can also take a threshold β of elements of δ in order to restrict the number of important features in the heatmap.

V. Explanation of Siamese neural networks

Let $S = \{(\mathbf{x}_i, \mathbf{x}_j, z_{ij}),\ (i,j) \in K\}$ be a dataset consisting of N pairs of feature vectors $\mathbf{x}_i \in \mathbb{R}^m$ and $\mathbf{x}_j \in \mathbb{R}^m$ such that a binary label $z_{ij} \in \{0, 1\}$ is assigned to every pair $(\mathbf{x}_i, \mathbf{x}_j)$. If both feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ are semantically similar, then z_{ij} takes value 0. If the vectors are semantically dissimilar, i.e., they correspond to different classes, then z_{ij} takes value 1. This implies that the training set S can be divided into two subsets: the similar subset labeled with $z_{ij} = 0$ and the dissimilar subset labeled with $z_{ij} = 1$. Knowledge of classes is not necessary if we have only weak information about similarity of pairs of examples. One of the models dealing with the weak information is the SNN [12]. Its architecture is shown in Fig. 4. If there are two semantically similar/dissimilar feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, then the contrastive loss function is used to train the SNN, which tries to make the Euclidean distance $d(\mathbf{h}_i, \mathbf{h}_j)$ between the corresponding embeddings (the SNN outputs) $\mathbf{h}_i \in \mathbb{R}^D$ and $\mathbf{h}_j \in \mathbb{R}^D$ to be as small/large as possible. In order to define the loss function, we introduce the following notation:

$$l_{ij} = \begin{cases} \|\mathbf{h}_i - \mathbf{h}_j\|_2^2, & z_{ij} = 0, \\ \max(0, \tau - \|\mathbf{h}_i - \mathbf{h}_j\|_2^2), & z_{ij} = 1, \end{cases}$$

where τ is a predefined threshold.

Then the contrastive loss function is of the form:

$$L_{\text{Siam}}(W) = \sum_{(i,j) \in K} l_{ij} + \mu R(W).$$

Here $R(W)$ is a regularization term; W is the matrix of the neural subnet parameters; μ is a hyper-parameter which controls the strength of the regularization.

Let $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{z} \in \mathbb{R}^m$ be a pair of examples for explaining. They have the corresponding hidden representations $\mathbf{h_x}$ and $\mathbf{h_z}$, respectively, obtained from the SNN. By applying the trained VAE, we can get two sets of output vectors $g(\mathbf{h})$ corresponding to inputs $\mathbf{h_x}$ and $\mathbf{h_z}$. The first set A contains vectors $\mathbf{a}_k$ generated in vicinity of $\mathbf{a} = g(\mathbf{h_x})$. The second set B consists of vectors $\mathbf{b}_k$ generated in vicinity of $\mathbf{b} = g(\mathbf{h_z})$. These sets can be viewed as sets of vectors in hyperspheres with centers $\mathbf{a}$ and $\mathbf{b}$, respectively. Generating the vectors from sets A and B, we realize the implicit perturbation procedure of the hidden representations.

Let us consider two cases when examples $\mathbf{x}$ and $\mathbf{z}$ are semantically similar (Case 1), and when they are semantically dissimilar (Case 2).

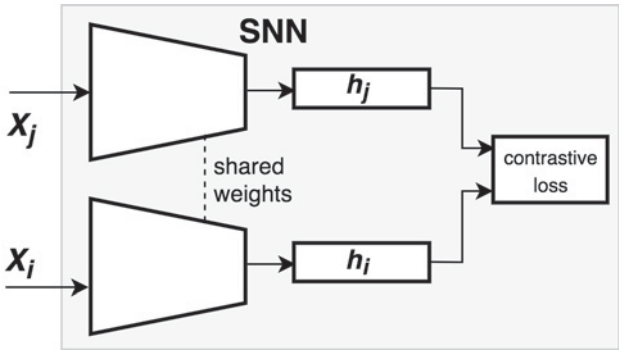

Fig. 4. The SNN architecture

Case 1: Semantic similarity assumes that the hidden representations $\mathbf{h_x}$ and $\mathbf{h_z}$ as well as **a** and **b** should be close to each other. This implies that we try to find their features which make them to be close. In other words, we are looking for features that contribute into a small distance between generated vectors and the vectors **a** or **b**. Hence, we have to select some number of generated vectors around **a**, which have smallest distances from vector **b**. These vectors characterize closeness of **a** to **b** and explain why **a** is semantically similar to **b**. Then images reconstructed from the set of generated vectors by means of the AE decoder indicate their features which change with changes of **a**. These changes produce the corresponding feature heatmap, explaining the similarity of **x** and **z**. Similarly, we select some number of generated vectors around **b**, which have smallest distances from vector **a**. These vectors characterize closeness of **b** to **a** and explain why **b** is semantically similar to **a**. Their reconstructed images for vectors around **b** produce the feature heatmap in the same way, explaining which parts of **z** are responsible for similarity with **x**.

Let us select N vectors $\mathbf{a}_k$ from the set A such that these vectors are nearest to vector **b** (the center of the set B). For every vector $\mathbf{a}_k$, we get the reconstructed image $\mathbf{x}_k^*$. In order to get the heatmap for image **x** explaining the similarity of images **x** and **z**, we compute the mean reconstructed image defined as

$$\mathbf{x}^{\#} = N^{-1} \sum_{k=1}^{N} \mathbf{x}_k^*.$$

Then the heatmap is computed as difference between the mean reconstructed image $\mathbf{x}^{\#}$ and the initial image **x**.

However, this is only a partial explanation of the similarity. In the same way, the heatmap for example **z** can be computed. We select N vectors $\mathbf{b}_k$ from the set B, which are nearest to vector **a** (the center of the set A). The reconstructed images $\mathbf{z}_k^*$ are obtained from vectors $\mathbf{b}_k$, and their mean vector is computed as

$$\mathbf{z}^{\#} = N^{-1} \sum_{k=1}^{N} \mathbf{z}_k^*.$$

The heatmap, explaining why the image **z** is semantically similar to **x**, is obtained as difference between $\mathbf{z}^{\#}$ and **z**.

Case 2: Semantic dissimilarity assumes that the hidden representations $\mathbf{h_x}$ and $\mathbf{h_z}$ as well as **a** and **b** should be as far as possible from each other. This implies that we try to find their features which make them far from each other. In contrast to Case 1, we select some number of generated vectors around **a**, which have largest distances from vector **b**. The next part of the algorithm is similar to Case 1.

VI. Numerical experiments

The proposed method is studied with the MNIST dataset (28×28 pixel handwritten digit images) and the CIFAR-10 dataset consisting of 32×32 color images drawn from 10 classes. The MNIST has a training set of 60,000 examples, and a test set of 10,000 examples. The CIFAR-10 dataset consists of 50,000 training and 10,000 test images each. It was collected by Krizhevsky et al. [55].

The length of the hidden representation layer is 20, i.e., the vector **h** consists of 20 features. We take $D = 20$, $M = 15$, $N_c = 5000$. The encoder of the AE is implemented in Keras by using 3 convolution and 3 max-pooling layers, one flatten layer and 2 dense layers with ReLU and Tanh (only the second dense layer) activation functions. The decoder is implemented by using 2 dense layers with ReLU functions, one reshape layer, 3 upsampling and 3 convolution layers with ReLU functions, but the last convolution layer uses sigmoid as activation function.

The encoder part of the VAE consists of 5 dense layers with dimension 50, one latent layer with dimension 35. The decoder part also has 5 dense layers with dimension 50 and one dense layer with dimension 15. All layers are implemented with the tanh activation functions. We also use a threshold β of the relative changes of features of $\mathbf{x}^*$ after perturbations of **a**, which is 0.5.

Triplets of pictures illustrating the proposed explanation method are shown in Fig.5 where the first picture in every row is an original image of a digit, the second picture is a heatmap of important features, and the third picture is the original image overlapped by the heatmap. The heatmaps show the important features which impact on the corresponding predictions of the black-box network. Fig.6 illustrates how digit 1 is explained when it incorrectly classified by the SNN as 3.

Similar triplets of pictures illustrating the proposed explanation method with using the CIFAR-10 dataset are shown in Fig. 7. They show examples of cats classified by the black-box model. It can be seen from the corresponding maps that they indicate on parts of pictures which actually explain cats. It is clearly seen in the third row of pictures where a typical furry tail is selected to explain the cat prediction.

VII. Conclusion

The main ideas behind the proposed method are to use the AE for reconstruction purposes and the VAE for linking the black-box output or outputs (for the SNN) with the AE code and implicit perturbing this code. The method is general and can be applied to various problems of the deep neural network explanation because the main ideas of the method do not depend on peculiarities of the black-box model.

One of the difficulties of the method is the possible lack of sufficient data for training the AE and the VAE. If we deal with the SNN, then this problem can be partially solved by using the corresponding training set consisting of concatenated pairs

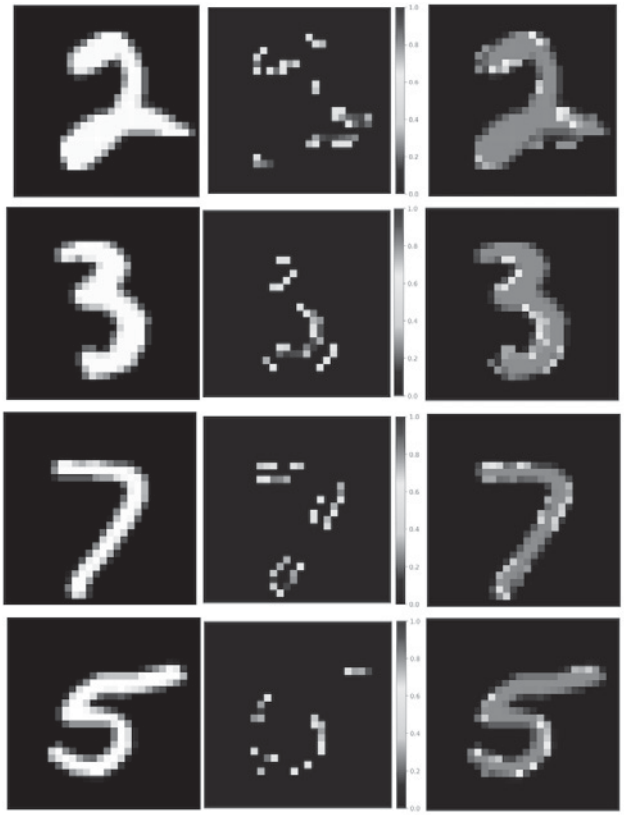

Fig. 5. Examples of explanation with digits 2, 3, 7, 5

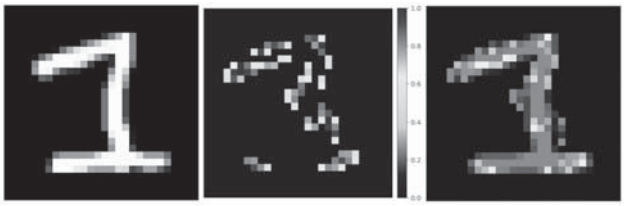

Fig. 6. An example of explanation with digit 1 incorrectly classified as 3

of original examples and concatenated pairs of embeddings. In this case, we can use the advantage of the SNN to significantly increase the dataset due to considering pairs of examples. Another way for improving the method in this case is to enlarge the AE code. This leads to the code consisting of two parts. The first part coincides with the black-box model output. The second part is auxiliary and used to get a better reconstruction of the original images. The corresponding numerical experiments and a detailed justification of the above improvements are directions for further research.

Acknowledgement

The reported study was funded by RFBR, project number 19-29-01004.

References

[1] A. Holzinger, G. Langs, H. Denk, K. Zatloukal, and H. Muller, "Causability and explainability of artificial intelligence in medicine," *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 4, p. e1312, 2019.

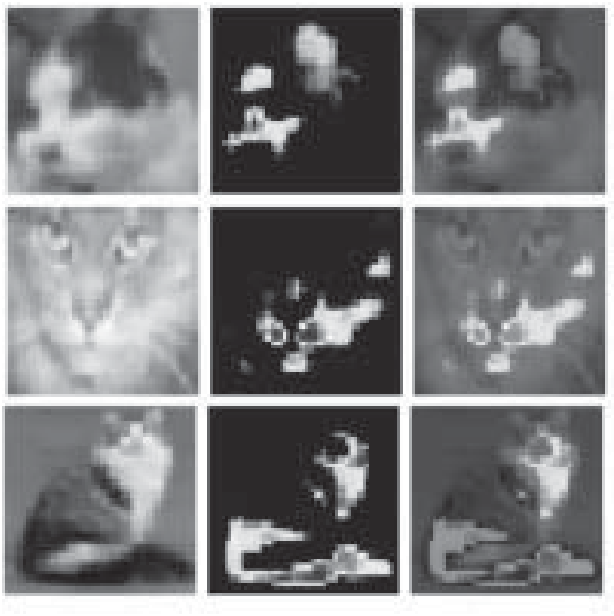

Fig. 7. Examples from the CIFAR-10 dataset

[2] V. Arya, R. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S. Hoffman, S. Houde, Q. Liao, R. Luss, A. Mojsilovic, S. Mourad, P. Pedemonte, R. Raghavendra, J. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. Varshney, D. Wei, and Y. Zhang, "One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques," Sep 2019, arXiv:1909.03012.

[3] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys*, vol. 51, no. 5, p. 93, 2019.

[4] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/: Published online, 2019.

[5] W. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yua, "Interpretable machine learning: definitions, methods, and applications," Jan 2019, arXiv:1901.04592.

[6] R. Fong and A. Vedaldi, "Explanations for attributing deep neural network predictions," in *Explainable AI*, ser. LNCS. Cham: Springer, 2019, vol. 11700, pp. 149–167.

[7] ——, "Interpretable explanations of black boxes by meaningful perturbation," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2017, pp. 3429–3437.

[8] M. Vu, T. Nguyen, N. Phan, and M. T. R. Gera, "Evaluating explainers via perturbation," Jun 2019, arXiv:1906.02032v1.

[9] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," May 2019, arXiv:1808.00033.

[10] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV 2014*, ser. LNCS, vol. 8689. Cham: Springer, 2014, pp. 818–833.

[11] J. Bromley, J. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Sackinger, and R. Shah, "Signature verification using a siamese time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 737–744, 1993.

[12] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546.

[13] D. Kingma and M. Welling, "Auto-encoding variational Bayes," May 2014, arXiv:1312.6114v10.

[14] M. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust You?" Explaining the predictions of any classifier," Aug 2016, arXiv:1602.04938v3.

[15] S. Shankaranarayana and D. Runje, "ALIME: Autoencoder based approach for local interpretability," Sep 2019, arXiv:1909.02437.

[16] I. Ahern, A. Noack, L. Guzman-Nateras, D. Dou, B. Li, and J. Huan, "NormLime: A new feature importance metric for explaining deep neural networks," Sep 2019, arXiv:1909.04200.

[17] M. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *AAAI Conference on Artificial Intelligence*, 2018, pp. 1527–1535.

[18] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, "GraphLIME: Local interpretable model explanations for graph neural networks," Jan. 2020, arXiv:2001.06216.

[19] D. Mardaoui and D. Garreau, "An analysis of lime for text data," Oct. 2020, arXiv:2010.12487.

[20] M. Kovalev, L. Utkin, and E. Kasimov, "SurvLIME: A method for explaining machine learning survival models," *Knowledge-Based Systems*, vol. 203, p. 106164, 2020.

[21] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[22] E. Strumbel and I. Kononenko, "An efficient explanation of individual classifications using game theory," *Journal of Machine Learning Research*, vol. 11, pp. 1–18, 2010.

[23] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GPDR," *Harvard Journal of Law & Technology*, vol. 31, pp. 841–887, 2017.

[24] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," Jun. 2018, arXiv:1806.07421.

[25] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, vol. 5, no. 4, pp. 2403–2424, 2011.

[26] B. Kim, C. Rudin, and J. Shah, "The Bayesian case model: A generative approach for case-based reasoning and prototype classification," in *Advances in Neural Information Processing Systems*, 2014, pp. 1952–1960.

[27] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[28] A. Arrieta, N. Diaz-Rodriguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," Oct. 2019, arXiv:1910.10045.

[29] D. Carvalho, E. Pereira, and J. Cardoso, "Machine learning interpretability: A survey on methods and metrics," *Electronics*, vol. 8, no. 832, pp. 1–34, 2019.

[30] A. Das and P. Rad, "Opportunities and challenges in explainableartificial intelligence (XAI): A survey," Jun. 2020, arXiv:2006.11371v2.

[31] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, pp. 206–215, 2019.

[32] N. Xie, G. Ras, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," Apr. 2020, arXiv:2004.14545.

[33] Z. Qi, S. Khorram, and F. Li, "Embedding deep networks into visual explanations," Apr. 2018, arXiv:1709.05360.

[34] L. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, "Generating visual explanations," in *European Conference on Computer Vision*. Cham: Springer, 2016, pp. 3–19.

[35] L. Hendricks, R. Hu, T. Darrell, and Z. Akata, "Grounding visual explanations," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 264–279.

[36] J. Rabold, H. Deininger, M. Siebers, and U. Schmid, "Enriching visual with verbal explanations for relational concepts: Combining LIME with Aleph," Oct. 2019, arXiv:1910.01837v1.

[37] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee, "Counterfactual visual explanations," Apr 2019, arXiv:1904.07451.

[38] A. Vellido, "The importance of interpretability and visualization in machine learning for applications in medicine and health care," *Neural Computing and Applications*, pp. 1–15, 2019.

[39] J. Wang, L. Gou, W. Zhang, H. Yang, and H. Shen, "DeepVID: Deep visual interpretation and diagnosis for image classifiers via knowledge distillation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 6, pp. 2168–2180, 2019.

[40] Q. Zhang and S.-C. Zhu, "Visual interpretability for deep learning: a survey," Feb. 2018, arXiv:1802.00614v2.

[41] V. Petsiuk, R. Jain, V. Manjunatha, V. Morariu, A. Mehra, V. Ordonez, and K. Saenko, "Black-box explanation of object detectors via saliency maps," Jun. 2020, arXiv:2006.03204.

[42] M. Mokuwe, M. Burke, and A. Bosman, "Black-box saliency map generation using bayesian optimisation," Jan. 2020, arXiv:2001.11366.

[43] S. Gulshad and A. Smeulders, "Explaining with counter visual attributes and examples," Jan. 2020, arXiv:2001.09671.

[44] L. Antwarg, R. Miller, B. Shapira, and L. Rokach, "Explaining anomalies detected by autoencoders using SHAP," Jun. 2020, arXiv:1903.02407v2.

[45] V. Bellini, A. S. andT. Di Noia, A. Ragone, and E. D. Sciascio, "Knowledge-aware autoencoders for explainable recommender systems," in *DLRS 2018: Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*, 2018, pp. 24–31.

[46] R. Guidotti, A. Monreale, S. Matwin, and D. Pedreschi, "Black box explanation by learning image exemplars in the latent feature space," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Cham: Springer, 2019, pp. 189–205.

[47] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders," 18 Nov 2015, arXiv:1511.05644v1.

[48] Z. Qi and F. Li, "Learning explainable embeddings for deep networks," in *NIPS Workshop on Interpretable Machine Learning*, vol. 31, 2017.

[49] P. Haghighi, O. Seton, and O. Nasraoui, "An explainable autoencoder for collaborative filtering recommendation," Dec. 2019, arXiv:2001.04344.

[50] A. Gee, D. G.-O. amd J. Ghosh, and D. Paydarfar, "Explaining deep classification of time-series data with learned prototypes," Apr. 2019, arXiv:1904.08935.

[51] C. Schockaert, V. Macher, and A. Schmitz, "VAE-LIME: Deep generative model based approach for local data-driven model interpretability applied to the ironmak ingindustry," Jul. 2020, arXiv:2007.10256.

[52] H. Uzunova, J. Ehrhardt, T. Kepp, and H. Handels, "Interpretable explanations of black box classifiers appliedon medical images by meaningful perturbations using variational autoencoders," in *Medical Imaging 2019: Image Processing*, vol. 10949. International Society for Optics and Photonics, 2019, p. 1094911.

[53] D. Alvarez-Melis and T. Jaakkola, "A causal framework for explaining the predictions of black-box sequence-to-sequence models," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 412–421.

[54] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[55] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Computer Science Department, University of Toronto, Tech. Rep. 1, 2009.