Computing and Informatics, Vol. 39, 2020, 1172–1202, doi: 10.31577/cai_2020_6_1172

EXPLANATION OF SIAMESE NEURAL NETWORKS FOR WEAKLY SUPERVISED LEARNING

Lev Utkin, Maxim Kovalev, Ernest Kasimov

Peter the Great Saint Petersburg Polytechnic University (SPbPU)
Saint Petersburg, Russia
e-mail: {lev.utkin, maxkovalev03, kasimov.ernest}@gmail.com

Abstract. A new method for explaining the Siamese neural network (SNN) as a black-box model for weakly supervised learning is proposed under condition that the output of every subnetwork of the SNN is a vector which is accessible. The main problem of the explanation is that the perturbation technique cannot be used directly for input instances because only their semantic similarity or dissimilarity is known. Moreover, there is no an "inverse" map between the SNN output vector and the corresponding input instance. Therefore, a special autoencoder is proposed, which takes into account the proximity of its hidden representation and the SNN outputs. Its pre-trained decoder part as well as the encoder are used to reconstruct original instances from the SNN perturbed output vectors. The important features of the explained instances are determined by averaging the corresponding changes of the reconstructed instances. Numerical experiments with synthetic data and with the well-known dataset MNIST illustrate the proposed method.

Keywords: Interpretable model, explainable AI, Siamese neural network, embedding, autoencoder, perturbation technique

Mathematics Subject Classification 2010: 68T10

1 INTRODUCTION

Machine learning models, especially, deep models play an important role in making prediction and decision for many applications. For example, computer-aided diagnosis systems using machine learning models for the diagnosis of various diseases have become a key element in medical imaging and personalized medicine over the

past few years. However, many modern efficient machine learning techniques are not easily explainable, they are black boxes, i.e., they do not explain their predictions in a way that humans could understand. This may be an obstacle for incorporating the machine learning models into applied areas like medicine. Often, humans are unable to effectively use predictions provided by the machine learning models without their interpretation and explanation. As a result, a lot of models have been developed to explain predictions of the deep classification and regression algorithms, for example, deep neural network predictions [1].

A clear taxonomy for understanding the diverse forms of explanations is provided by Arya et al. [2]. Since a lot of machine learning models are black boxes, then we mainly consider the so-called post-hoc explanations which involve auxiliary models to explain the black-box models after it has been trained. These auxiliary or explanation models can also be divided into three types: example-based, local and global models. According to the example-based approach, an instance from the training set is selected, which could explain the behavior of a black-box model. One of the methods implementing the example-based explanations is nearest neighbors [3]. Local or prediction-level models focus on explaining how they make individual predictions, namely, what features lead to the individual prediction (see [4] for a definition). Explanations are derived by fitting an interpretable model locally around the considered instance. Global or dataset-level models explain importance of features across a dataset or a population [5]. They explain the global relationships between features and the model predictions.

A key component of general explanations for the local models as well as for the global models is the contribution of individual input features. A prediction is computationally explained by assigning to each feature a number which denotes its impact on the prediction.

One of the important machine learning tasks is to compare pairs of objects, for example, pairs of images, pairs of data vectors, etc. The task can be solved in the framework of the distance metric learning approach [6, 7, 8], which is based on computing a corresponding pairwise metric function that measures a distance between data vectors or a similarity between the vectors. The basic idea behind the metric learning solution is that the distance between similar objects should be smaller than the distance between different objects.

A powerful implementation of the metric learning dealing with non-linear data structures is the so-called Siamese neural network (SNN) [9, 10]. 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. In a simplest case, when the training set is labelled, i.e., every instance belongs to a class, then two instances are similar if they belong to the same class. Two instances are dissimilar if they belong to different classes. At the same time, the SNN can be applied to the weakly supervised case when there are no labels of training instances, but there is only a side infor-

mation of relationship of instances, i.e., of semantic similarity or dissimilarity of instances.

An approach to explain the Siamese neural network (SNN) as a black-box model is proposed. The approach is agnostic to the black-box model. This means that we do not know or do not use any details of the black-box model. Only its input and the corresponding output are used for training the explanation model. We also assume that the explained SNN is regarded as the black-box model in the sense that we know input feature vectors and embedded vectors, but we do not know peculiarities and tuning parameters of the SNN.

At the same time, to the best of our knowledge, there are no appropriate algorithms for explaining the SNN which is used as a weakly supervised learning model. Therefore, we aim to solve this explanation problem and to propose the corresponding approach which allows us to get subsets of features explaining similarity and dissimilarity of certain instances.

First of all, let us consider main difficulties of explaining the SNN. They are the following:

1. The input vectors (instances) are semantically similar or dissimilar. Therefore, direct distances between the input feature vectors do not have a sense. Semantically dissimilar vectors may be closer to each other than semantically similar vectors.
2. The perturbation technique cannot be used directly for input instances of the SNN due to the possible large dimensionality of input data.
3. Prototypes cannot be defined in case of weakly supervised data. Moreover, prototypes defined as mean values of the input vectors also do not have a sense because the input vectors may be too different. They can be defined only by using the output vectors (embeddings).
4. There is no an "inverse" map between the embedded vectors and the corresponding input feature vectors, i.e., we do not know subsets of features in the input vector corresponding to some features of the embedded vector.

Taking into account the fact that the distance measurement has a sense only for the SNN output vectors, direct ideas for explaining the SNN are the following:

1. to select a predefined number of features from a pair of embeddings of semantically similar instances, which have the smallest distance between each other;
2. to select a predefined number of features from a pair of embeddings of semantically dissimilar instances, which have the largest Euclidean distance between each other;
3. to perturb the selected features in a specific way;
4. to reconstruct the obtained perturbation in order to find which features of reconstructed instances are changed in accordance with the perturbation of embeddings;

5. maximally changed features of reconstructed instances explain similarity or dissimilarity of the considered original instances.

These ideas could be viewed as an approach for developing a SNN explanation method, but a bottleneck of the approach is the reconstruction of instances from the corresponding embeddings. Our experiments have shown that it is very difficult to train a reconstruction neural network, having the embedded vectors as its input and the original instances as its output, due to a large difference between the instance dimensionality and the embedded vector size. Therefore, we propose to apply an autoencoder (AE) of a special type in order to overcome the above difficulty and to help for reconstructing the instances from their embeddings. The proposed AE is trained in a way different from the standard AE. In contrast to the standard AE, it takes into account the proximity of its hidden representation and the SNN outputs by means of extending the corresponding loss function. The decoder part of the pretrained AE can be regarded as the reconstruction neural network after its additional training by using the SNN outputs. Our numerical experiments have demonstrated that this scheme allows us to reconstruct images (instances) with a high accuracy. By perturbing the SNN outputs in a special way and using the trained decoder of the AE, we can consider how these perturbations impact on the reconstructed instances.

The paper is organized as follows. Related work concerning with available explanation models and the SNN applications is considered in Section 2. A detailed description of the SNN architecture is given in Section 3. Main ideas of the proposed SNN explanation method are discussed in Section 4. Details of the AE implementation for improving the reconstruction of original images from the corresponding embeddings are given in Section 5. A scheme of the used perturbation technique and the instance reconstruction is studied in Section 6. Numerical experiments illustrating the proposed method on the basis of the synthetic dataset and the well-known MNIST dataset are given in Section 7. Concluding remarks are provided in Section 8.

2 RELATED WORK

2.1 Explanation Models

There are a lot of approaches to locally explain black-box models. One of the very popular methods is the Local Interpretable Model-agnostic Explanations (LIME) [11]. The main intuition of LIME is that the explanation may be derived locally from a set of synthetic instances generated randomly in the neighborhood of the instance to be explained such that every synthetic instance has a weight according to its proximity to the explained instance. Several modifications of LIME have been proposed due to success and simplicity of the method, for example, ALIME [12], NormLIME [13], DLIME [14], Anchor LIME [15], LIME-SUP [16], LIME-Aleph [17], SurvLIME [18]. Garreau and Luxburg [19] proposed a thorough theoretical analysis of LIME.

Another very popular method is the SHAP [20] and its modifications which take a game-theoretic approach by optimizing a regression loss function based on Shapley values [21, 22, 23, 24]. Alternative methods are influence functions [25], a multiple hypothesis testing framework [26], and many other methods.

A large part of methods can be united as counterfactual explanations [27]. They try to answer the question: "Why is the outcome Y obtained instead of Z?". A simplest approach to answer the question is to find the nearest training instance belonging to another class. As shown by Moore et al. [28], this approach strongly depends on the size and quality of the considered training set, and it cannot find a counterfactual that is not explicitly in the set. Therefore, a lot of new methods of the counterfactual explanation have been developed [29, 30, 31, 32, 33, 34, 35].

Dhurandhar et al. [36, 37] extended the counterfactual explanation by introducing the so-called contrastive explanation methods which produce explanations of the form [36]: "An input feature vector is classified in class y because features $f_{i_1}, \ldots, f_{i_k}$ are present and because features $f_{j_1}, \ldots, f_{j_l}$ are absent".

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 [38, 39]. The explanation methods using this technique determine how an explained instance is similar to a prototype.

It is important to point out also that most aforementioned explanation methods starting from LIME [11] are based on perturbation technique [40, 41, 42, 43, 44, 45]. These methods assume that contribution of a feature can be determined by measuring how prediction score changes when the feature is altered [46]. Strumbel and Kononenko [20] propose to perturb the input feature vectors and to observe how changes of the input features correspond to changes of the outcome. They rely on an assumption that a feature is important and strongly impacts the outcome if its change sufficiently changes the outcome. Perturbation techniques are model-agnostic, i.e., perturbations can be applied to a black-box model without any need to access the internal structure of the model. However, the perturbation technique may meet computational difficulties when perturbed input instances have a lot of features.

Several comprehensive surveys devoted to various explainable methods can be found in literature [47, 48, 49, 1, 50]. A very interesting critical review of main assumptions and statements accepted in explaining the black-box machine learning models is provided by Rudin [51]. The review [51] considers in detail how to avoid mistakes and incorrect assumptions in explanation models. The corresponding software packages [52] are developed in order to simplify the explanation process for various machine learning models.

2.2 Metric Learning and Siamese Neural Networks

A detailed description of the metric learning approaches is represented by Le Capitaine [53] and by Kulis [7]. One of the most important and popular approaches

is to use the Mahalanobis distance as a distance metric which assumes some linear structure of data. However, this assumption significantly restricts the applicability of the Mahalanobis distance for comparing pairs of objects. Therefore, in order to overcome this restriction, the kernelization of linear methods is one of the possible ways for solving the metric learning problem. Bellet et al. [6] review several approaches and algorithms to deal with nonlinear forms of metrics. In particular, these are the support vector metric learning algorithm provided by Xu et al. [54], the gradient-boosted large margin nearest neighbors method proposed by Kedem et al. [55], the Hamming distance metric learning algorithm provided by Norouzi et al. [56]. Various methods and applications of the metric learning can be found in [57, 58, 59, 60, 61, 62, 54, 63].

SNNs realize a non-linear embedding of data [64] and were introduced in 90s by Bromley and LeCun to solve signature verification as an image matching problem [9]. SNNs have been widely spread in solving many application problems. In particular, they are applied to problems of image recognition and verification [10, 65, 66, 67, 68, 69], of speaker verification [70], of visual tracking [71, 72], of novelty and anomaly detection [73, 74], and to many different theoretical and practical problems [75, 76, 77, 78]. An important application of the SNN is one-shot learning [79] or few-shot learning [80, 81, 82, 83], when it is supposed that there are only a few training instances in some classes for training. A more detailed and general definition of the one-shot and few-shot learning is given in [84].

It should be noted that the above applications present only a small part of all applications of the SNNs. Many modifications of SNNs have been also developed, including fully-convolutional SNNs [85], SNNs combined with a gradient boosting classifier [86], SNNs with the triangular similarity metric [8].

3 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: a similar or positive set with $z_{ij} = 0$ and a dissimilar or negative set with $z_{ij} = 1$. It should be noted that knowledge of classes is not necessary if we have only weak information about similarity of pairs of instances.

The main idea of using the SNN can be formulated as follows. If there are two feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ being dissimilar, then the Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j)$ between these feature vectors should be as large as possible. However, this may be not a case in practice. For instance, if to consider the medicine application, then tuberculosis and adenocarcinoma in the lung cancer diagnosis may have similar computed tomography patterns, but these are quite different diseases. Adenocarcinoma

is cancer, but tuberculosis is not. At the same time, different forms of lung cancer may look quite differently, for instance, lepidic and squamous cell carcinoma. In this case, the Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j)$ may be rather large, but it should be as small as possible. In other words, the Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j)$ often does not correspond to semantic similarity of objects. Therefore, we have to consider not the distance between feature vectors themselves, but the distance between new feature representations or embeddings denoted as $\mathbf{h}_i = (h_1^{(i)}, \ldots, h_D^{(i)}) \in \mathbb{R}^D$ and $\mathbf{h}_j = (h_1^{(j)}, \ldots, h_D^{(j)}) \in \mathbb{R}^D$, which fulfil the conditions of distances and similarity, i.e., the Euclidean distance $d(\mathbf{h}_i, \mathbf{h}_j)$ between vectors $\mathbf{h}_i$ and $\mathbf{h}_j$ should be as small (large) as possible for a pair of objects with $z_{ij} = 0$ ($z_{ij} = 1$). At that, every vector $\mathbf{h}_i$ is a map f of $\mathbf{x}_i$ to a low-dimensional space, i.e., $\mathbf{h}_i = f(\mathbf{x}_i)$ and $\mathbf{h}_j = f(\mathbf{x}_j)$. The function f is implemented by every subnetwork in the SNN.

A standard architecture of the SNN given in the literature (see, for example, [10]) is shown in Figure 1. It consists of two identical neural subnets which are trained to compare a pair of feature vectors in terms of their semantic similarity or dissimilarity.

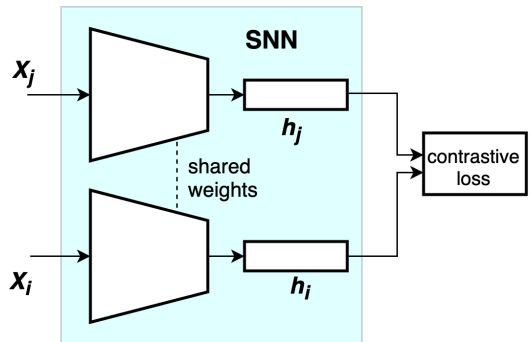


Figure 1. The architecture of the SNN

It should be noted that there are many specific loss function for training the SNN [6, 53, 8], which solve the problem of the object comparison. One of the functions is the contrastive loss function defined as

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

where τ is a predefined threshold.

Hence, the total error function for minimizing is defined as

$$L_{\text{Siam}}(W) = \sum_{(i,j) \in K} l(\mathbf{x}_i, \mathbf{x}_j, z_{ij}) + \mu R(W). \tag{2}$$

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

It is assumed below that the outputs $\mathbf{h}_i$ and $\mathbf{h}_j$ are known for every pair $(\mathbf{x}_i, \mathbf{x}_j)$.

4 A GENERAL IDEA FOR EXPLAINING THE SNN

First of all, we have to define what is the meaning of the semantically similar and dissimilar instances from the interpretation point of view. In other words, we have to explain why two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ are semantically similar or dissimilar, i.e., which features of the instances make them similar or dissimilar. This is not a trivial question because the similarity of two instances and the distance between them in the input space may be not correlated. Therefore, it is difficult to apply various techniques, for example perturbation schemes, to the input examples. However, the similarity and the distance can be considered in the output space, where the distance between embeddings $\mathbf{h}_i$ and $\mathbf{h}_j$ corresponding to similar examples is supposed to be rather small, and the distance between embeddings corresponding to dissimilar examples is large. This implies that there are features of the vectors $\mathbf{h}_i$ and $\mathbf{h}_j$, which determine the similarity of input instances by comparing the corresponding distances between these features. These features can be called important features.

In order to explain the important features defining the semantic similarity and dissimilarity, we consider a simple example. Suppose that embeddings consist of two features h_1 and h_2, i.e., $D = 2$. Three 2-dimensional vectors are shown in Figure 2 in the form of small circles and a triangle. It can be seen from Figure 2 that the first and the second points correspond to semantically similar instances because they are close to each other. It is obvious that the first and the third points are semantically dissimilar because they are far from each other. The semantic similarity of points 1 and 2 is defined by feature h_2 because distance d_{similar} is smallest. This implies that the second feature can be viewed as important for semantic similarity of two instances. It should be noted that points 1 and 2 are close to each other also due to feature h_1. However, its impact is smaller in comparison with the impact of feature h_2. We can say the same about the first and the third points. They are dissimilar due to feature h_1 because the large distance $d_{\text{dissimilar}}$ defines the semantic dissimilarity of the instances. This implies that the first feature can be viewed as important for semantic dissimilarity of instances 1 and 3.

In sum, we have a rule for determining important features of embeddings, which define semantic similarity and dissimilarity of input examples. The main problem is that we do not know how these important features of embeddings are connected with the corresponding original instances because there is no an "inverse" map from embeddings to input instances. Therefore, in order to solve the interpretation problem, we have to construct this "inverse" map and to find features of the input instances which correspond to important features of embeddings. If we had such the "in-

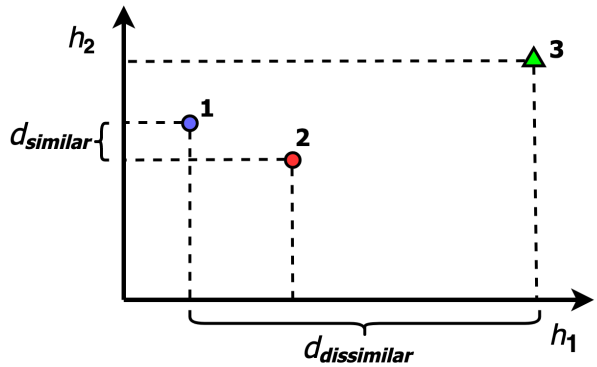


Figure 2. Pairs of semantically similar (1 and 2) and dissimilar (1 and 3) instances whose embeddings consist of two features (h_1 and h_2)

verse" map, then the problem would be solved by perturbing the important features of embeddings and analyzing the corresponding changes of the input instances.

One of the ways for implementing the "inverse" map is to train a reconstruction neural network with embeddings $\mathbf{h}$ as inputs and the corresponding input instances $\mathbf{x}$ as outputs. However, our experiments have shown that this reconstruction network is overfitted and does not allow to correctly reconstruct the input instances especially when the number of training instances is not large and the instances $\mathbf{x}$ are images of a high dimensionality. It turns out that it is simpler to train an AE and then to use its pre-trained decoder part for additional training and for reconstruction. Therefore, our idea is to train the AE whose inputs are instances $\mathbf{x}$, its code (the hidden representation) is close to the embedding vector $\mathbf{h}$. The trained decoder part of the AE can be used as a pre-trained reconstruction neural network which transforms embeddings $\mathbf{h}$ into instances $\mathbf{x}$. Moreover, this reconstruction neural network can be additionally trained by using embeddings $\mathbf{h}$ and instances $\mathbf{x}$.

Having important features of vectors $\mathbf{h}_i$ and $\mathbf{h}_j$ and the trained reconstruction neural network, we perturb the features in accordance with the following rules. If the pair of similar instances is analyzed, then the features are perturbed to reduce the distance between these important features. The perturbation of important features of dissimilar instances is carried out to increase the distance between them. Changes of the reconstructed instances produce the corresponding heatmaps explaining similarity or dissimilarity of two input instances.

Perturbations aim at describing how the output of the explained model changes when one or more input features are perturbed. The intuition of the technique is that the more a model's response depend on a feature, the more predictions or some output score change with the corresponding feature changes. The perturbation scheme can be regarded as one of the interesting approaches to the model interpre-

tation development and to the explainer evaluation [44]. Suppose there is a feature vector $\mathbf{x} \in \mathbb{R}^m$ that is slightly perturbed to a new vector $\mathbf{x} + \delta$, where δ is a small perturbation that does not alter the meaning of the data point, i.e., the vector $\mathbf{x} + \delta$ remains the similarity relationship with other vectors from the training set without changes. The perturbation scheme can be also viewed in the framework of a sensitivity analysis method which aims to consider how the output of the explainable model changes when one or more input features are perturbed. Perturbation methods have the advantages of a straightforward interpretation, as they are a direct measure of the marginal effect of some input features to the output [40]. Moreover, perturbation schemes can be simply implemented, and they can be applied to post-hoc models.

Finally, the proposed method for explaining the SNN can be represented by means of an algorithm consisting of two parts. The first part aims to train the additional AE with a special loss function, which plays a partial role of the explainer. It aims to reconstruct the input instances from the training set and to take into account the SNN output. The second part is to train the decoder of the pre-trained AE, to perturb the embedding vectors at the SNN output, to use the decoder in order to reconstruct the perturbed embeddings and to observe the features of the reconstructed vectors which are changed due to the perturbation of embeddings.

Let us consider every part of the above algorithm in detail. Suppose that we have a trained SNN as a black-box model. For every input instance $\mathbf{x}_i$, we have the corresponding embedding vector $\mathbf{h}_i \in \mathbb{R}^D$ such that $\mathbf{h}_i = f(\mathbf{x}_i)$. For the sake of clarity, we will call instances $\mathbf{x}_i$ as images whereas the embeddings will be called as vectors.

5 PRE-TRAINING OF THE AE

Suppose that inputs of the proposed AE are images $\mathbf{x}_i$. Then we expect to get reconstructed images $\mathbf{x}_i^*$ as its outputs. The corresponding loss function $L_{\text{recon\_AE}}$ for training the AE is defined, for example, as follows:

$$L_{\text{recon\_AE}}(W, \mathbf{x}_i, \mathbf{x}_i^*) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}_i^*\|_2^2 \,. \tag{3}$$

A regularization term is not written here because it will be used later. In order to use the pre-trained decoder for reconstruction of vector $\mathbf{h}_i \in \mathbb{R}^D$, the AE has to be trained in a special way. First of all, the length of a part of its hidden representation has to coincide with the length of vector $\mathbf{h}$, which is equal to D. Second, the loss function should take into account proximity of vectors $\mathbf{h}_i$ and the corresponding vectors of the AE hidden representation denoted as $\mathbf{b}_i \in \mathbb{R}^D$, i.e., we need to have the vectors $\mathbf{b}_i$ in the hidden layer coinciding with the vectors $\mathbf{h}_i$ obtained by means of the SNN. Therefore, we propose to change the loss function for training the AE by adding the loss function L_{close} in the following way:

$$L_{\text{autoen}}(W) = \gamma L_{\text{recon\_AE}}(W, \mathbf{x}_i, \mathbf{x}_i^*) + \mu L_{\text{close}}(W, \mathbf{h}_i, \mathbf{b}_i) + \lambda R(W)$$
$$= \gamma \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}_i^*\|_2^2 + \mu \sum_{i=1}^{n} \|\mathbf{h}_i - \mathbf{b}_i\|_2^2 + \lambda R(W). \quad (4)$$

Here $R(W)$ is a regularization term, λ is a hyper-parameter which controls the strength of the regularization; W is the set of the neural network weights; γ and μ are parameters that control the interaction of the loss function terms.

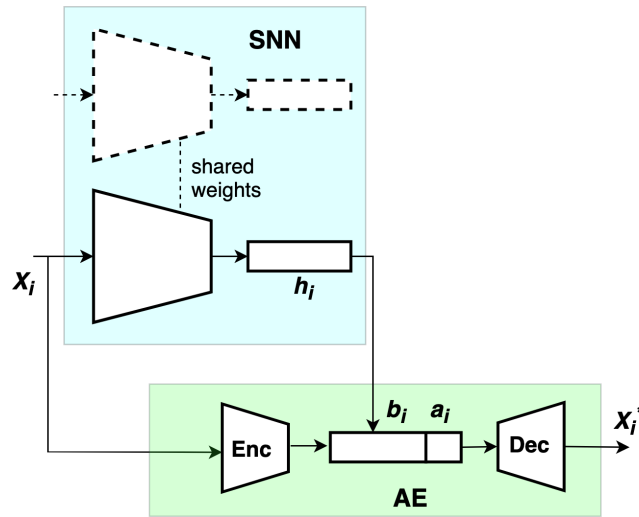


Figure 3. The autoencoder training scheme

One of the problems here is a case when D is small. Hence, the AE may provide the unsatisfactory reconstruction if the hidden representation is of a too small size. Therefore, for improving the presented architecture, it is proposed to enlarge the hidden representation of the AE, i.e., the vector $\mathbf{b}_i$ by means of its concatenation with a vector $\mathbf{a}_i \in \mathbb{R}^A$. As a result, we get the vector $\mathbf{c}_i = (\mathbf{b}_i||\mathbf{a}_i) \in \mathbb{R}^{D+A}$, where $(\mathbf{b}_i||\mathbf{a}_i)$ denotes the concatenation operation of vectors $\mathbf{b}_i$ and $\mathbf{a}_i$. The enlarged embedding allows us to improve the decoder training. In the same way, we later enlarge the outputs of the SNN, which contain vectors $(\mathbf{h}_i||\mathbf{a}_i) \in \mathbb{R}^{D+A}$ of the same dimensionality. Before training the AE, we assume that the vector $\mathbf{a}_i$ concatenated with the SNN output is arbitrary, for example, with zero-valued elements.

A scheme of the first part of the explanation algorithm is shown in Figure 3. It can be seen from the scheme that the AE is trained by using embedding vectors $\mathbf{h}_i$ from subnetworks of the SNN and the input images $\mathbf{x}_i$.

The pre-trained decoder part as well as the trained encoder part of the AE can be used for additional training and for reconstruction of the perturbed embeddings that is for implementing the second part of the algorithm. The use of the AE allows us to significantly simplify the training process and to get acceptable vector reconstructions. It should be noted that an architecture of the encoder differs from the architecture of a subnetwork of the SNN because we consider the SNN as a black box whose architecture is unknown.

6 PERTURBATION OF EMBEDDINGS AND THE INSTANCE RECONSTRUCTION

The second part of the explanation algorithm, including the perturbation and the image reconstruction is shown in Figure 4.

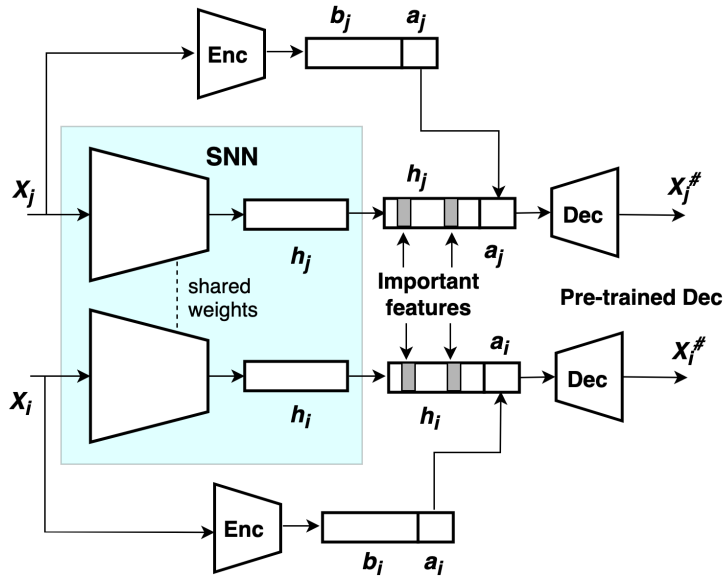


Figure 4. A scheme of the second part of the explanation algorithm

The pre-trained decoder can be again trained by using only vectors $\mathbf{h}_i$ from the SNN output and vectors $\mathbf{a}_i$ from the AE encoder output. The concatenated vector $(\mathbf{h}_i||\mathbf{a}_i)$ is the decoder input, the reconstructed image $\mathbf{x}_i^\#$ is the decoder output. The loss function is the standard Euclidean distance between images $\mathbf{x}_i^\#$ and $\mathbf{x}_i$. It is important to note that the vector $\mathbf{a}_i$ is computed by means of the encoder part of the AE. The set of all vectors $\mathbf{a}_i$ corresponding to all training instances could be separately stored in order to avoid the repeated use of the encoder. However, this can be done only for training. When we have new instances, the encoder has to be used. If the SNN outputs are not so small, then the AE hidden representation

can have the same size as vectors $\mathbf{h}_i$. In this case, $A = 0$ and vectors $\mathbf{a}_i$ as well as the encoder are not needed, and the scheme of the second part of the explanation algorithm is simplified.

For a new pair of images $\mathbf{x}_i$ and $\mathbf{x}_j$, we find vectors $\mathbf{h}_i$ and $\mathbf{h}_j$ as outputs of the SNN. If the images are semantically similar, then we find a predefined number of important features in $\mathbf{h}_i$ and $\mathbf{h}_j$ with smallest distances. There are different ways for choosing important features. The first way is just to define the number of important features $s < D$ such that the index set $J \subseteq \{1, \ldots, D\}$ consists of s indices corresponding to smallest distances between features $h_k^{(i)}$ and $h_k^{(j)}$, $k = 1, .., D$. In this case, the value of s can be regarded as a tuning parameter. The second way is to define a threshold α of the relative Euclidean distances $r_k \in [0, 1]$ between important features of two vectors $\mathbf{h}_i$ and $\mathbf{h}_j$. It is supposed that features are important if there holds

$$r_k = \frac{\left|h_k^{(i)} - h_k^{(j)}\right|}{\max_{l=1,\ldots,D}\left|h_l^{(i)} - h_l^{(j)}\right|} \leq \alpha, k = 1, \ldots, D. \tag{5}$$

Then the index set J of important features is defined as

$$J = \{k : r_k \leq \alpha\}. \tag{6}$$

In the same way, we define the rule for important features of the semantically dissimilar images. In particular, features in $\mathbf{h}_i$ and $\mathbf{h}_j$ with largest distances can be viewed ad important features.

By having a set of important features, we can perturb them to study how the important features of the SNN outputs impact on the original images $\mathbf{x}_i$ and $\mathbf{x}_j$. The trained decoder is used to reconstruct images from $\mathbf{h}_i + \delta_i$ and $\mathbf{h}_j + \delta_j$ and to investigate how features of the reconstructed images $\mathbf{x}_i^{\#}$ and $\mathbf{x}_j^{\#}$ are changed. Here δ_i and δ_j are the perturbation vectors such that indices of their non-zero elements are from the index set J, other elements are equal to zero. In sum, we have the embeddings $\mathbf{h}_i$ and $\mathbf{h}_j$, the reconstructed images $\mathbf{x}_i^{\#}$ and $\mathbf{x}_j^{\#}$, the index set J of important features of embeddings. Important features as an example (two features) in $\mathbf{h}_i$ and $\mathbf{h}_j$ are shown by dashed cells in Figure 4. The perturbed vectors $\mathbf{h}_i$ and $\mathbf{h}_j$ are fed to the corresponding decoders in order to get the reconstructed images $\mathbf{x}_i^{\#}$ and $\mathbf{x}_j^{\#}$ which depends on perturbations.

Suppose that the perturbation δ of an embedding leads to the changed i^{th} feature $x_i^{\#}(\delta)$ of the reconstructed image $\mathbf{x}^{\#}$. After generating the random vector δ many times, say N times, the mean value of the i^{th} feature changes is defined as

$$T_i = N^{-1} \sum_{j=1}^{N} \left(x_i^{\#}(\delta_j) - x_i^{\#} \right). \tag{7}$$

It should be noted that T_i may be positive as well as negative. If we consider the visual interpretation, then all values of T_i are scaled to be in interval $[-1, 1]$.

Finally, we compute absolute values T_i^* of T_i in order to visualize the largest changes. The heatmaps explaining the considered instances are determined from condition $T_i^* \geq \beta$, i.e., they have largest changes of the reconstructed instance. Here β is a threshold of relative changes of features, which can be regarded as another tuning parameter.

The perturbation vectors are randomly generated in the following way. Suppose that the index set J consists of s elements. First, we generate the vector δ in the s-sphere defined as $B = \{\delta \in R^s : |\delta| = R\}$ with some predefined radius R. There are several methods for the uniform sampling of points δ in the s-sphere with the unit radius $R = 1$, for example, [87, 88]. Then every generated point is multiplied by R. Moreover, we take vectors δ only from a part of the s-sphere. This part is defined by a hyper-quadrant, which the second embedding $\mathbf{h}_j$ is located in, under condition that the vector $\mathbf{h}_i$ is in the origin of coordinates. The radius is defined as a portion of the Euclidean distance $d(\mathbf{h}_j, \mathbf{h}_i)$ between $\mathbf{h}_j$ and $\mathbf{h}_i$, i.e., $R = q \cdot d(\mathbf{h}_j, \mathbf{h}_i)$, where q is also a tuning parameter.

7 NUMERICAL EXPERIMENTS

7.1 A Synthetic Example

One of the questions of the SNN explanation is to understand how the instances may be semantically similar or dissimilar and how to explain the important features of the instances, which are responsible for the semantics. This question is not trivial. Indeed, we simply and logically understand the similarity or dissimilarity at the embedding level because they depend on the distance between vectors. However, the similarity or dissimilarity of images are semantic and, therefore, not obvious. In order to see what the similarity and dissimilarity mean for images, we consider a synthetic example. We consider two types of randomly generated images: triangles and circles. Sizes of triangles and circles may be different and random. All images are of 28×28 pixels.

The SNN as well as the AE are implemented in Python by using the Keras package with Tensorflow. They are trained on the generated set of images with circles and triangles. The length of the hidden representation layer of the AE is 41, i.e., vector $\mathbf{h}$ consists of 32 features ($D = 32$), and vector $\mathbf{a}$ is of the length 9. The hyper-parameter μ, which controls the strength of the regularization is 0.02. The loss function for training the SNN is of the form:

$$l(\mathbf{x}_i, \mathbf{x}_j, z_{ij}) = \begin{cases} \left(\|\mathbf{h}_i - \mathbf{h}_j\|_2^2 - \omega\right)^2, & z_{ij} = 0, \\ \left(\max(0, \tau - \|\mathbf{h}_i - \mathbf{h}_j\|_2^2)\right)^2, & z_{ij} = 1. \end{cases} \tag{8}$$

Here ω is an intraclass margin which is introduced to diverse instances of the same class. It can be seen from (8) that the loss function differs from (1). The function (8) is used because embeddings of similar instances are too close to each

other. A simple way to make them different is to introduce the margin ω which sets the minimal distance between embeddings equal to ω.

Numerical results are shown in Figures 5, 6, 7, 8 such that every figure contains numerical results of 6 experiments depending on the value of important features s for perturbation. Every experiment is represented by means of four pictures or two pairs of pictures. The first pair (vertically) consists of original images, whose similarity or dissimilarity has to be explained, overlapped by the corresponding heatmaps. The second pair of pictures is the heatmaps of features explaining the similarity or dissimilarity. It is important to note that the heatmaps are obtained by using the pre-trained autoencoder.

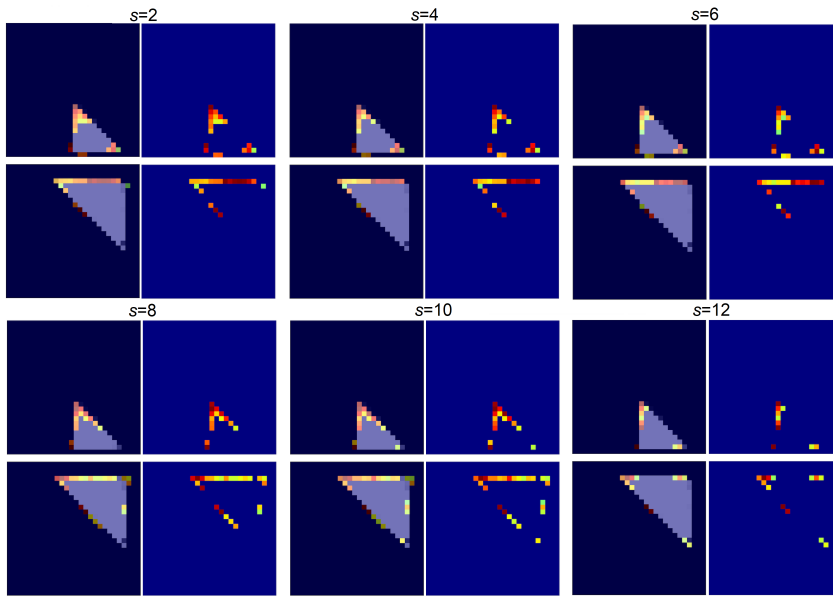


Figure 5. Examples of semantically similar images (triangles) and the explanation heatmaps for $s = 2, 4, 6, 8, 10, 12$

It can be seen from Figure 5 that the semantic similarity of triangles is defined starting from angles of the triangles (see the corresponding pictures of heatmaps by $s = 2$). This is a very interesting result. Indeed, the triangles are intuitively similar by angles. Sides of triangles also play some role in explanation, but they may be like some parts of small circles due to the low resolution of images. It can be seen from pictures by increasing the value of $s = 4, 6, 8$. We get sides of triangles as important features. It is interesting to note that insides of the triangles almost do not participate in the explanation though these are solid on the original images and differ from the background color. It is also interesting to see that that increasing of s does not necessarily leads to increasing the important features of the reconstructed images. For example, the number of important features by $s = 12$ is less than by

$s = 10$. This is due to the fact that some new perturbations mask changes the previous perturbations.

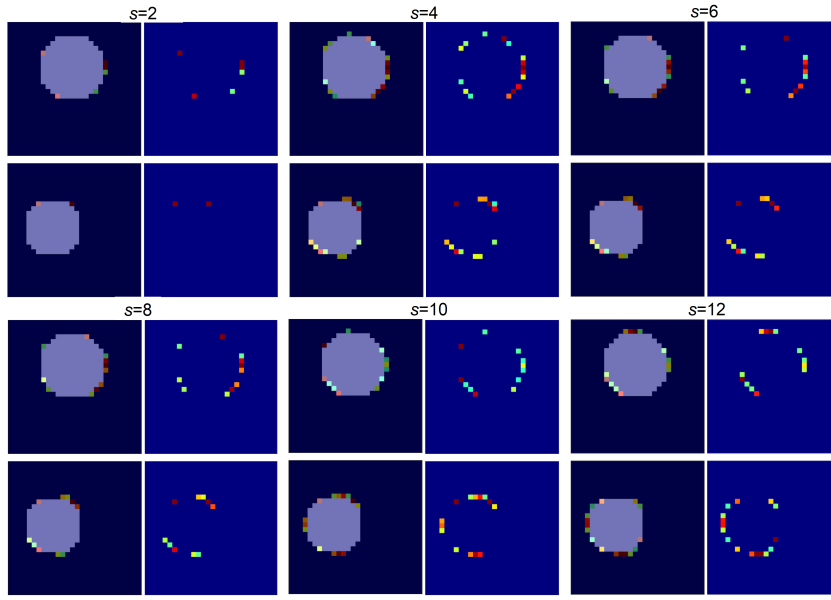


Figure 6. Examples of semantically similar images (circles) and the explanation heatmaps for $s = 2, 4, 6, 8, 10, 12$

When we look at Figure 6, it can be seen that the circles are like rectangles with chamferings due to the low resolution. It is also an interesting case, because the similarity is observed mainly in chamferings, and it covers sides only partially by increasing s.

But the most interesting cases are shown in Figures 7 and 8, where the semantic dissimilarity is studied. Two cases are studied:

1. The original triangle is small and the circle is large.
2. The original triangle is large and the circle is small.

It is clearly seen from the first pictures that the dissimilarity is explained by angles of triangles and by chamferings of circles. Indeed, triangles as well as "rectangles-circles" have sides as similar elements. They do not explain the semantic dissimilarity. Only angles of triangles and chamferings are really different. The sides of triangles and the whole circles become important only by large values of s. We intentionally consider the images of the low resolution in order to see some peculiarities of explaining the "wrong" rectangle and the "wrong" circle. It is again important to point out that insides of the pictures do not participate in the explanation.

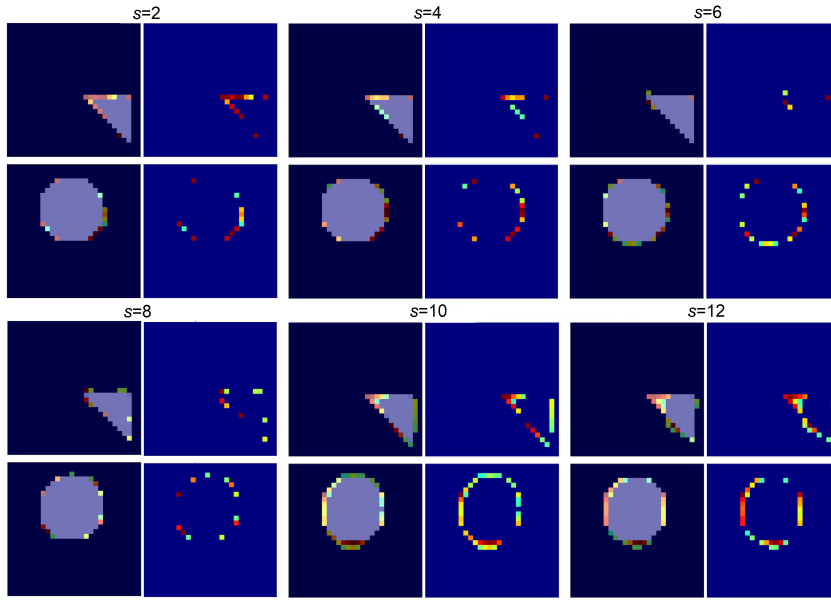


Figure 7. Examples of semantically dissimilar images (large circles and small triangles) and the explanation heatmaps for $s = 2, 4, 6, 8, 10, 12$

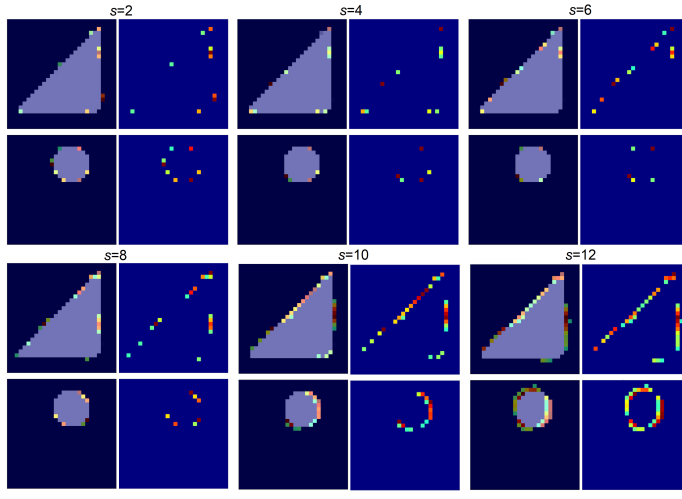


Figure 8. Examples of semantically dissimilar images (small circles and large triangles) and the explanation heatmaps for $s = 2, 4, 6, 8, 10, 12$

Architectures of the SNN and the AE are not provided for these numerical examples because they are stated to understand the meaning of the similarity and dissimilarity explanations. However, the corresponding architectures for explaining results obtained on the MNIST dataset are given below.

7.2 MNIST

The proposed explanation method is studied by applying the MNIST dataset which is a commonly used large dataset of 28x28 pixel handwritten digit images [89]. It has a training set of 60 000 instances, and a test set of 10 000 instances. The digits are size-normalized and centered in a fixed-size image. The dataset is available at `http://yann.lecun.com/exdb/mnist/`. If two digits from the MNIST dataset are identical, i.e., they belong to the same class, then they are semantically similar. If two digits are different, then they are semantically dissimilar.

7.3 Architecture of Neural Networks

Layer	Dimension	Activation
Input	28×28	–
Conv1	$28 \times 28 \times 4$	ReLU
Pooling1	$14 \times 14 \times 4$	–
Flatten	784	–
Dense1	128	ReLU
Output (Dense2)	20	Tanh

Table 1. An architecture of every subnetwork of the SNN for the MNIST dataset

Layer	Dimension	Activation
Input	28×28	–
Conv1	$28 \times 28 \times 8$	ReLU
Pooling1	$14 \times 14 \times 8$	–
Conv2	$14 \times 14 \times 16$	ReLU
Pooling2	$7 \times 7 \times 16$	–
Flatten	784	–
Hidden	26	Tanh
Dense	784	ReLU
Reshape	$7 \times 7 \times 16$	–
Up-Sampling	$14 \times 14 \times 16$	–
Conv3	$14 \times 14 \times 8$	ReLU
Up-Sampling	$28 \times 28 \times 8$	–
Output (Conv4)	20	Sigmoid

Table 2. An architecture of the AE for the MNIST dataset

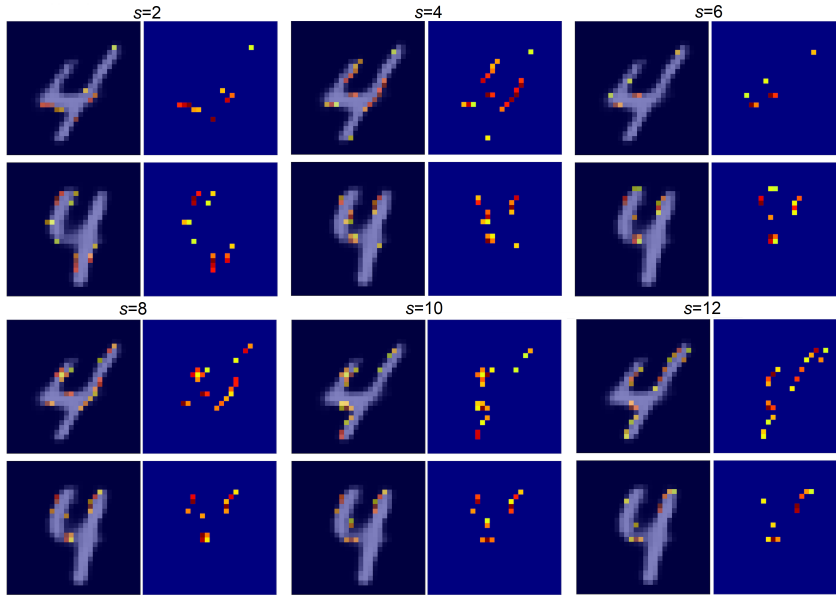


Figure 9. Examples of semantically similar images of digits 4 from the MNIST dataset and the explanation heatmaps for $s = 2, 4, 6, 8, 10, 12$

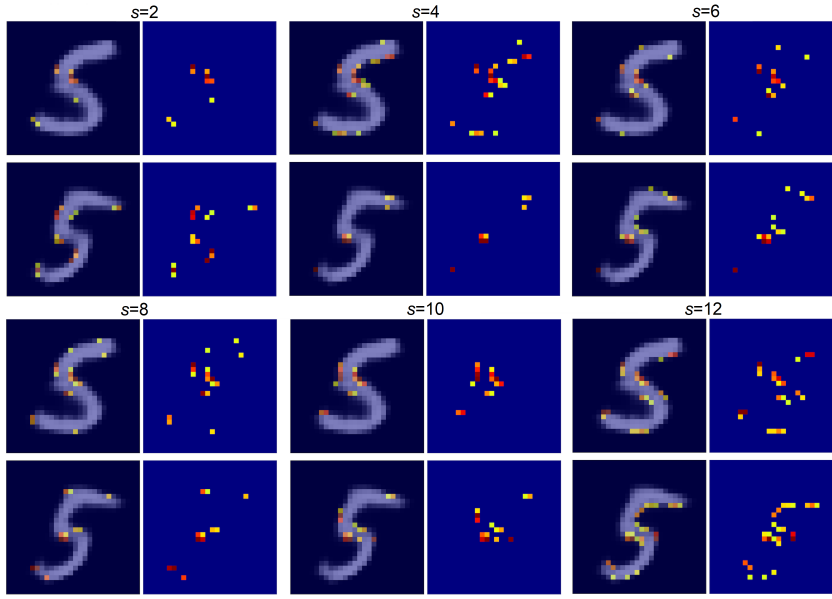


Figure 10. Examples of semantically similar images of digits 5 from the MNIST dataset and the explanation heatmaps for $s = 2, 4, 6, 8, 10, 12$

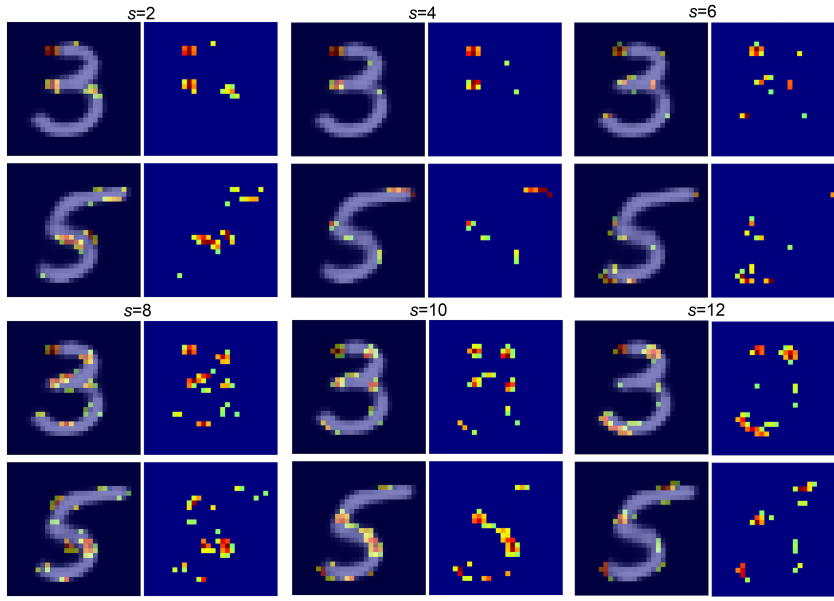


Figure 11. Examples of semantically dissimilar images of digits 3 and 5 from the MNIST dataset and the explanation heatmaps for $s = 2, 4, 6, 8, 10, 12$

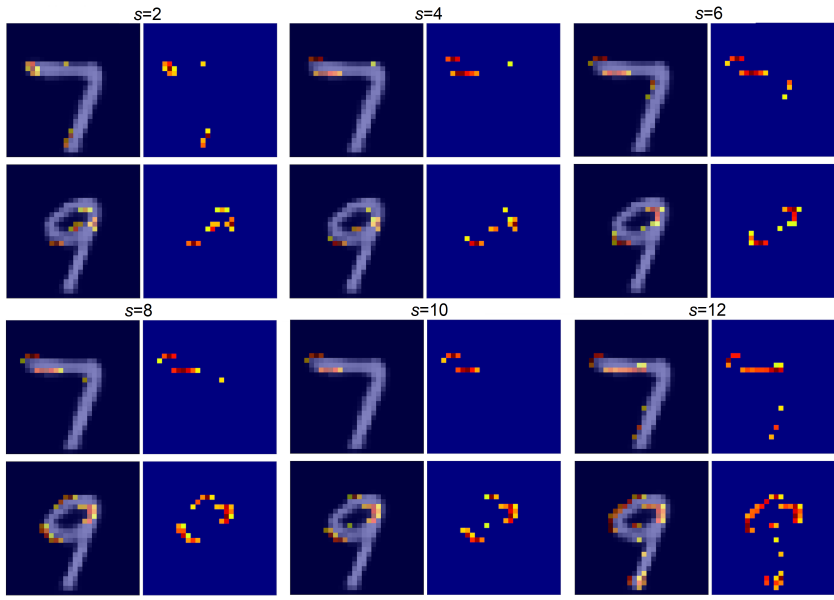


Figure 12. Examples of semantically dissimilar images of digits 7 and 9 from the MNIST dataset and the explanation heatmaps for $s = 2, 4, 6, 8, 10, 12$

An architecture of the SNN for experiments with the MNIST dataset is shown in Table 1. Every subnetwork is implemented as a convolutional network having a convolutional layer (Conv1), a max pooling layer (Pooling1), flatten layer (Flatten), which transforms a two-dimensional matrix into a vector, two dense layers (Dense1 and Dense2), which are represented by fully connected networks. Parameters of the loss function (8) are $\mu = 0.0005$, $\omega = 0.5$, $\tau = 3$.

An architecture of the AE is shown in Table 2. The encoder part consists of two convolutional layers (Conv1 and Conv2), two max pooling layers (Pooling1 and Pooling2), flatten layer (Flatten). The decoder part consists of a dense layer (Dense), a reshape layer to change dimensions (Reshape), two upsampling layers and two convolutional layers (Conv3 and Conv4).

The length of the hidden representation layer is 26, i.e., vector $\mathbf{h}$ consists of 20 features ($D = 20$), and vector $\mathbf{a}$ is of the length 6.

7.4 Results

Numerical results illustrating the explanation method on the MNIST dataset are shown in Figures 9, 10, 11, 12. The figures have the same structure as Figures 5, 6, 7, 8, i.e., every figure contains numerical results of 6 experiments depending on the value of important features s for perturbation.

Figures 9 and 10 show semantically similar digits 4 and 5, respectively. It can be seen from pictures in Figure 9 that the selected features indicate peculiarities of the digit 4, which differ from other digits. In particular, the important features are located at the upper part of the digit and its middle part. The same can be said about the digit 5 shown in Figure 10. The important features are concentrated at the middle part of the digits and partially select their upper curve. We again see from Figures 9 and 10 that the semantic similarity of pairs of original images is clearly exhibited by means of important features which explain this similarity.

Figures 11 and 12 show semantically dissimilar digits 3,5 and 7,9, respectively. The explanation of the semantic dissimilarity is very explicit in Figure 11. Indeed, the selected important features are inherent in the difference of two digits. Figure 12 is also very demonstrative. It can be seen from Figure 12 that only the parts of digits 7 and 9 are selected for explanation that characterize differences between the different digits. It is interesting to note that the slanting vertical slashes are not selected for explanation because they are common for these two digits. They are only partly selected when $s = 12$, i.e., features of embeddings responsible for the semantic similarity begin to act.

It can be also concluded from all the considered examples that the choice of a proper value of parameter s is not a trivial task. It can be solved by enumerating several values s and depends on many factors: datasets, the length of embeddings, the dimensionality of images, etc.

8 CONCLUSION

A new method for explaining the SNN results under condition of the weakly supervised learning has been presented in this paper. Basic ideas behind the method are comparisons of the explained instances at the embedding level and reconstruction of the embedding feature vectors by means of the separately trained decoder and the encoder of the AE in order to analyze the impact of the embedding vector perturbations on the reconstructed features of original instances. It should be noted that the main elements of the proposed method such as the AE with the extended loss function and the perturbation technique can be implemented independently of a structure of the explained SNN. This implies that the proposed method can be applied to various applications using SNNs. The method can be also used when there is information about classes of training data. This case can be viewed as a special case of the considered explanation approach.

To the best of our knowledge, there are no explanation methods applied to the SNN. Therefore, we do not compare the proposed method with the well-known methods such as LIME, SHAP, etc.

One of the limitations of the proposed approach is a possible small amount of training data in order to train the AE. The SNN is trained on pairs of instances such that the number of pairs may be large even by a small number of original instances. However, the AE has to be trained only by using the available data. One of the ways to overcome this problem is to train the AE on concatenated pairs of original instances. However, our experiments have shown that this obvious way may lead to the unsatisfactory reconstruction. Therefore, a modification of the method taking into account the lack of the sufficient amount of training data is a direction for further research. Another limitation is a rather large number of tuning parameters, including the number of important features, the length of the AE hidden representation, etc. Some efficient rules for restricting their values can be also regarded as a direction for further research.

It is important to note that the idea to reconstruct original instances from embeddings by means of the AE with the modified loss function can be applied not only to SNNs, but to different neural networks which implement the feature extraction procedures. If we have information about output feature extracted vectors of the neural networks, then the corresponding results can be explained in the same way. The main difference is that prototypes of classes should be used in order to select the important extracted features instead of output vectors of the SNN. In other words, in order to explain an original instance, the corresponding feature extracted vector is compared with the prototype of the class of this instance, and the nearest features are selected for their perturbation. In the same way, the counterfactual explanation technique can be applied by considering the prototypes of other classes. A detailed study of these extensions of the proposed method is another direction for further research.

Acknowledgement

The reported study was funded by RFBR, project No. 20-01-00154.

REFERENCES

[1] Guidotti, R.—Monreale, A.—Ruggieri, S.—Turini, F.—Giannotti, F.—Pedreschi, D.: A Survey of Methods for Explaining Black Box Models. ACM Computing Surveys, Vol. 51, 2019, No. 5, Art. No. 93, doi: 10.1145/3236009.

[2] Arya, V.—Bellamy, R. K. E.—Chen, P. Y.—Dhurandhar, A.—Hind, M.—Hoffman, S. C.—Houde, S.—Liao, Q. V.—Luss, R.—Mojsilović, A.—Mourad, S.—Pedemonte, P.—Raghavendra, R.—Richards, J.—Sattigeri, P.—Shanmugam, K.—Singh, M.—Varshney, K. R.—Wei, D.—Zhang, Y.: One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques. arXiv:1909.03012, 2019.

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

[4] Murdoch, W. J.—Singh, C.—Kumbier, K.—Abbasi-Asl, R.—Yu, B.: Interpretable Machine Learning: Definitions, Methods, and Applications. PNAS, Vol. 116, 2019, No. 44, pp. 22071–22080, doi: 10.1073/pnas.1900654116.

[5] Ibrahim, M.—Louie, M.—Modarres, C.—Paisley, J. W.: Global Explanations of Neural Networks: Mapping the Landscape of Predictions. Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (AIES '19), 2019, pp. 279–287, doi: 10.1145/3306618.3314230.

[6] Bellet, A.—Habrard, A.—Sebban, M.: A Survey on Metric Learning for Feature Vectors and Structured Data. arXiv:1306.6709, 2013.

[7] Kulis, B.: Metric Learning: A Survey. Foundations and Trends in Machine Learning, Vol. 5, 2013, No. 4, pp. 287–364, doi: 10.1007/s11042-015-2847-3.

[8] Zheng, L.—Duffner, S.—Idrissi, K.—Garcia, C.—Baskurt, A.: Siamese Multi-Layer Perceptrons for Dimensionality Reduction and Face Identification. Multimedia Tools and Applications, Vol. 75, 2016, No. 9, pp. 5055–5073, doi: 10.1007/s11042-015-2847-3.

[9] Bromley, J.—Bentz, J.—Bottou, L.—Guyon, I.—LeCun, Y.—Moore, C.—Sackinger, E.—Shah, R.: Signature Verification Using a "Siamese" Time Delay Neural Network. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, 1993, No. 4, pp. 669–688, doi: 10.1142/S0218001493000339.

[10] Chopra, S.—Hadsell, R.—LeCun, Y.: Learning a Similarity Metric Discriminatively, with Application to Face Verification. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, 2005, pp. 539–546, doi: 10.1109/CVPR.2005.202.

[11] Ribeiro, M.—Singh, S.—Guestrin, C.: "Why Should I Trust You?" Explaining the Predictions of Any Classifier. arXiv:1602.04938v3, 2016.

[12] Shankaranarayana, S. M.—Runje, D.: ALIME: Autoencoder Based Approach for Local Interpretability. In: Yin, H., Camacho, D., Tino, P., Tallón-Ballesteros, A., Menezes, R., Allmendinger, R. (Eds.): Intelligent Data Engineering and Automated Learning – IDEAL 2019. Springer, Cham, Lecture Notes in Computer Science, Vol. 11871, 2019, pp. 454–463, doi: 10.1007/978-3-030-33607-3_49.

[13] Ahern, I.—Noack, A.—Guzmán-Nateras, L.—Dou, D.—Li, B.—Huan, J.: NormLime: A New Feature Importance Metric for Explaining Deep Neural Networks. arXiv:1909.04200, 2019.

[14] Zafar, M. R.—Khan, N. M.: DLIME: A Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems. arXiv:1906.10263, 2019.

[15] Ribeiro, M. T.—Singh, S.—Guestrin, C.: Anchors: High-Precision Model-Agnostic Explanations. AAAI Conference on Artificial Intelligence, 2018, pp. 1527–1535.

[16] Hu, L.—Chen, J.—Nair, V. N.—Sudjianto, A.: Locally Interpretable Models and Effects Based on Supervised Partitioning (LIME-SUP). arXiv:1806.00663, 2018.

[17] Rabold, J.—Deininger, H.—Siebers, M.—Schmid, U.: Enriching Visual with Verbal Explanations for Relational Concepts – Combining LIME with Aleph. In: Cellier, P., Driessens, K. (Eds.): Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2019). Springer, Cham, Communications in Computer and Information Science, Vol. 1167, 2019, pp. 180–192, doi: 10.1007/978-3-030-43823-4_16.

[18] Kovalev, M. S.—Utkin, L. V.—Kasimov, E. M.: SurvLIME: A Method for Explaining Machine Learning Survival Models. Knowledge-Based Systems, Vol. 203, 2020, Art. No. 106164, doi: 10.1016/j.knosys.2020.106164.

[19] Garreau, D.—von Luxburg, U.: Explaining the Explainer: A First Theoretical Analysis of LIME. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR, Vol. 108, 2020, pp. 1287–1296.

[20] Štrumbelj, E.—Kononenko, I.: An Efficient Explanation of Individual Classifications Using Game Theory. Journal of Machine Learning Research, Vol. 11, 2010, pp. 1–18.

[21] Aas, K.—Jullum, M.—Løland, A.: Explaining Individual Predictions When Features Are Dependent: More Accurate Approximations to Shapley Values. arXiv:1903.10464, 2019.

[22] Ancona, M.—Oztireli, C.—Gross, M.: Explaining Deep Neural Networks with a Polynomial Time Algorithm for Shapley Values Approximation. Proceedings of the 36th International Conference on Machine Learning, PMLR, Vol. 97, 2019, pp. 272–281.

[23] Lundberg, S. M.—Lee, S. I.: A Unified Approach to Interpreting Model Predictions. In: Guzon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 4765–4774.

[24] Owen, A. B.—Prieur, C.: On Shapley Value for Measuring Importance of Dependent Inputs. SIAM/ASA Journal on Uncertainty Quantification, Vol. 5, 2017, pp. 986–1002, doi: 10.1137/16M1097717.

[25] Koh, P. W.—Liang, P.: Understanding Black-Box Predictions via Influence Functions. Proceedings of the 34th International Conference on Machine Learning, PMLR, Vol. 70, 2017, pp. 1885–1894.

[26] Burns, C.—Thomason, J.—Tansey, W.: Interpreting Black Box Models with Statistical Guarantees. arXiv:1904.00045, 2019.

[27] Wachter, S.—Mittelstadt, B.—Russell, C.: Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. Harvard Journal of Law and Technology, Vol. 31, 2018, pp. 841–887, doi: 10.2139/ssrn.3063289.

[28] Moore, J.—Hammerla, N.—Watkins, C.: Explaining Deep Learning Models with Constrained Adversarial Examples. In: Nayak, A., Sharma, A. (Eds.): PRICAI 2019: Trends in Artificial Intelligence. Springer, Cham, Lecture Notes in Computer Science, Vol. 11670, 2019, pp. 43–56, doi: 10.1007/978-3-030-29908-8_4.

[29] Goyal, Y.—Wu, Z.—Ernst, J.—Batra, D.—Parikh, D.—Lee, S.: Counterfactual Visual Explanations. Proceedings of the 36th International Conference on Machine Learning, PMLR, Vol. 97, 2019, pp. 2376–2384.

[30] Hendricks, L. A.—Hu, R.—Darrell, T.—Akata, Z.: Grounding Visual Explanations. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.): Computer Vision – ECCV 2018. Springer, Cham, Lecture Notes in Computer Science, Vol. 11206, 2018, pp. 269–286, doi: 10.1007/978-3-030-01216-8_17.

[31] Laugel, T.—Lesot, M.-J.—Marsala, C.—Renard, X.—Detyniecki, M.: Comparison-Based Inverse Classification for Interpretability in Machine Learning. In: Medina, J. et al. (Eds.): Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations (IPMU 2018). Springer, Cham, Communications in Computer and Information Science, Vol. 853, 2018, pp. 100–111, doi: 10.1007/978-3-319-91473-2_9.

[32] Liu, S.—Kailkhura, B.—Loveland, D.—Han, Y.: Generative Counterfactual Introspection for Explainable Deep Learning. 2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2019, doi: 10.1109/globalsip45357.2019.8969491.

[33] Van Looveren, A.—Klaise, J.: Interpretable Counterfactual Explanations Guided by Prototypes. arXiv:1907.02584, 2019.

[34] Poyiadzi, R.—Sokol, K.—Santos-Rodríguez, R.—De Bie, T.—Flach, P. A.: FACE: Feasible and Actionable Counterfactual Explanations. Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES '20), 2020, pp. 344–350, doi: 10.1145/3375627.3375850.

[35] van der Waa, J.—Robeer, M.—van Diggelen, J.—Brinkhuis, M.—Neerincx, M.: Contrastive Explanations with Local Foil Trees. arXiv:1806.07470, 2018.

[36] Dhurandhar, A.—Chen, P. Y.—Luss, R.—Tu, C. C.—Ting, P.—Shanmugam, K.—Das, P.: Explanations Based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. arXiv:1802.07623v2, 2018.

[37] Dhurandhar, A.—Pedapati, T.—Balakrishnan, A.—Chen, P. Y.—Shanmugam, K.—Puri, R.: Model Agnostic Contrastive Explanations for Structured Data. arXiv:1906.00117, 2019.

[38] Bien, J.—Tibshirani, R.: Prototype Selection for Interpretable Classification. The Annals of Applied Statistics, Vol. 5, 2011, No. 4, pp. 2403–2424, doi: 10.1214/11-AOAS495.

[39] Kim, B.—Rudin, C.—Shah, J.: The Bayesian Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification. In: Ghahramani, Y., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 27 (NIPS 2014), 2014, pp. 1952–1960.

[40] Ancona, M.—Ceolini, E.—Öztireli, C.—Gross, M.: Gradient-Based Attribution Methods. In: Samek, W., Montavon, G., Vedaldi, A., Hansen, L., Müller, K. R. (Eds.): Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Springer, Cham, Lecture Notes in Computer Science, Vol. 11700, 2019, pp. 169–191, doi: 10.1007/978-3-030-28954-6_9.

[41] Fong, R.—Vedaldi, A.: Explanations for Attributing Deep Neural Network Predictions. In: Samek, W., Montavon, G., Vedaldi, A., Hansen, L., Müller, K. R. (Eds.): Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Springer, Cham, Lecture Notes in Computer Science, Vol. 11700, 2019, pp. 149–167, doi: 10.1007/978-3-030-28954-6_8.

[42] Fong, R.—Vedaldi, A.: Interpretable Explanations of Black Boxes by Meaningful Perturbation. Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3449–3457, doi: 10.1109/ICCV.2017.371.

[43] Petsiuk, V.—Das, A.—Saenko, K.: RISE: Randomized Input Sampling for Explanation of Black-Box Models. arXiv:1806.07421, 2018.

[44] Vu, M. N.—Nguyen, T. D.—Phan, N.—Gera, R.—Thai, M. T.: Evaluating Explainers via Perturbation. arXiv:1906.02032v1, 2019.

[45] Zeiler, M. D.—Fergus, R.: Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.): Computer Vision – ECCV 2014. Springer, Cham, Lecture Notes in Computer Science, Vol. 8689, 2014, pp. 818–833, doi: 10.1007/978-3-319-10590-1_53.

[46] Du, M.—Liu, N.—Hu, X.: Techniques for Interpretable Machine Learning. Communications of the ACM, Vol. 63, 2019, No. 1, pp. 68–77, doi: 10.1145/3359786.

[47] Adadi, A.—Berrada, M.: Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). IEEE Access, Vol. 6, 2018, pp. 52138–52160, doi: 10.1109/ACCESS.2018.2870052.

[48] Arrieta, A. B.—Díaz-Rodríguez, N.—Del Ser, J.—Bennetot, A.—Tabik, S.—Barbado, A.—Garcia, S.—Gil-Lopez, S.—Molina, D.—Benjamins, R.—Chatila, R.—Herrera, F.: Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI. Information Fusion, Vol. 58, 2020, pp. 82–115, doi: 10.1016/j.inffus.2019.12.012.

[49] Gilpin, L. H.—Bau, D.—Yuan, B. Z.—Bajwa, A.—Specter, M.—Kagal, L.: Explaining Explanations: An Overview of Interpretability of Machine Learning. 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), 2018, pp. 80–89, doi: 10.1109/DSAA.2018.00018.

[50] Mohseni, S.—Zarei, N.—Ragan, E. D.: A Survey of Evaluation Methods and Measures for Interpretable Machine Learning. arXiv:1811.11839v1, 2018.

[51] Rudin, C.: Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. Nature Machine Intelligence, Vol. 1, 2019, pp. 206–215, doi: 10.1038/s42256-019-0048-x.

[52] Nori, H.—Jenkins, S.—Koch, P.—Caruana, R.: InterpretML: A Unified Framework for Machine Learning Interpretability. arXiv:1909.09223, 2019.

[53] Le Capitaine, H.: Constraint Selection in Metric Learning. arXiv:1612.04853v1, 2016.

[54] Xu, Z.—Weinberger, K. Q.—Chapelle, O.: Distance Metric Learning for Kernel Machines. arXiv:1208.3422, 2012.

[55] Kedem, D.—Tyree, S.—Sha, F.—Lanckriet, G.—Weinberger, K.: Non-Linear Metric Learning. In: Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 25 (NIPS 2012), 2012, pp. 2582–2590.

[56] Norouzi, M.—Fleet, D. J.—Salakhutdinov, R. R.: Hamming Distance Metric Learning. In: Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 25 (NIPS 2012), 2012, pp. 1070–1078.

[57] Hoffer, E.—Ailon, N.: Deep Metric Learning Using Triplet Network. In: Feragen, A., Pelillo, M., Loog, M. (Eds.): Similarity-Based Pattern Recognition (SIMBAD 2015). Springer, Cham, Lecture Notes in Computer Science, Vol. 9370, 2015, pp. 84–92, doi: 10.1007/978-3-319-24261-3_7.

[58] Huang, K.—Jin, R.—Xu, Z.—Liu, C. L.: Robust Metric Learning by Smooth Optimization. Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI 2010), 2010, pp. 244–251.

[59] Li, C.—Georgiopoulos, M.—Anagnostopoulos, G. C.: Kernel-Based Distance Metric Learning in the Output Space. The 2013 International Joint Conference on Neural Networks (IJCNN), IEEE, 2013, pp. 1–8, doi: 10.1109/IJCNN.2013.6706862.

[60] Schultz, M.—Joachims, T.: Learning a Distance Metric from Relative Comparisons. In: Thrun, S., Saul, L., Schölkopf, B. (Eds.): Advances in Neural Information Processing Systems 16 (NIPS 2003), 2003, pp. 41–48.

[61] Weinberger, K. Q.—Saul, L. K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. Journal of Machine Learning Research, Vol. 10, 2009, pp. 207–244.

[62] Xing, E.—Jordan, M.—Russell, S.—Ng, A.: Distance Metric Learning with Application to Clustering with Side-Information. In: Becker, S., Thrun, S., Obermayer, K. (Eds.): Advances in Neural Information Processing Systems 15 (NIPS 2002), 2002, pp. 505–512.

[63] Yin, X.—Chen, Q.: Deep Metric Learning Autoencoder for Nonlinear Temporal Alignment of Human Motion. 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 2160–2166, doi: 10.1109/ICRA.2016.7487366.

[64] Roy, S.—Harandi, M.—Nock, R.—Hartley, R.: Siamese Networks: The Tale of Two Manifolds. Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Vol. 2, 2019, pp. 3046–3055, doi: 10.1109/ICCV.2019.00314.

[65] He, K.—Zhang, X.—Ren, S.—Sun, J.: Deep Residual Learning for Image Recognition. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[66] Hu, J.—Lu, J.—Tan, Y. P.: Discriminative Deep Metric Learning for Face Verification in the Wild. 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1875–1882, doi: 10.1109/CVPR.2014.242.

[67] Sun, Y.—Chen, Y.—Wang, X.—Tang, X.: Deep Learning Face Representation by Joint Identification-Verification. In: Ghahramani, Y., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 27 (NIPS 2014), 2014, pp. 1988–1996.

[68] Yi, D.—Lei, Z.—Liao, S.—Li, S. Z.: Deep Metric Learning for Person Re-Identification. Proceedings of the 2014 22nd International Conference on Pattern Recognition (ICPR), 2014, pp. 34–39, doi: 10.1109/ICPR.2014.16.

[69] Zhang, C.—Liu, W.—Ma, H.—Fu, H.: Siamese Neural Network Based Gait Recognition for Human Identification. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 2832–2836, doi: 10.1109/ICASSP.2016.7472194.

[70] Chen, K.—Salman, A.: Extracting Speaker-Specific Information with a Regularized Siamese Deep Network. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 24 (NIPS 2011), 2011, pp. 298–306.

[71] Jiang, C.—Xiao, J.—Xie, Y.—Tillo, T.—Huang, K.: Siamese Network Ensemble for Visual Tracking. Neurocomputing, Vol. 275, 2018, pp. 2892–2903, doi: 10.1016/j.neucom.2017.10.043.

[72] Zhan, H.—Ni, W.—Yan, W.—Wu, J.—Bian, H.—Xiang, D.: Visual Tracking Using Siamese Convolutional Neural Network with Region Proposal and Domain Specific Updating. Neurocomputing, Vol. 275, 2018, pp. 2645–2655, doi: 10.1016/j.neucom.2017.11.050.

[73] Masana, M.—Ruiz, I.—Serrat, J.—van de Weijer, J.—Lopez, A. M.: Metric Learning for Novelty and Anomaly Detection. arXiv:1808.05492, 2018.

[74] Utkin, L. V.—Zaborovsky, V. S.—Lukashin, A. A.—Popov, S. G.—Podolskaja, A. V.: A Siamese Autoencoder Preserving Distances for Anomaly Detection in Multi-Robot Systems. 2017 International Conference on Control, Artificial Intelligence, Robotics and Optimization (ICCAIRO), Prague, Czech Republic, IEEE, 2017, pp. 39–44, doi: 10.1109/ICCAIRO.2017.17.

[75] Berlemont, S.—Lefebvre, G.—Duffner, S.—Garcia, C.: Class-Balanced Siamese Neural Networks. Neurocomputing, Vol. 273, 2018, pp. 47–56, doi: 10.1016/j.neucom.2017.07.060.

[76] Dhami, D. S.—Kunapuli, G.—Page, D.—Natarajan, S.: Predicting Drug-Drug Interactions from Molecular Structure Images. AAAI Fall Symposium – AI for Social Good, AAAI, 2019, pp. 1–6.

[77] Shaham, U.—Lederman, R. R.: Learning by Coincidence: Siamese Networks and Common Variable Learning. Pattern Recognition, Vol. 74, 2018, pp. 52–63, doi: 10.1016/j.patcog.2017.09.015.

[78] Wang, J.—Fang, Z.—Lang, N.—Yuan, H.—Su, M. Y.—Baldi, P.: A Multi-Resolution Approach for Spinal Metastasis Detection Using Deep Siamese Neural Networks. Computers in Biology and Medicine, Vol. 84, 2017, pp. 137–146, doi: 10.1016/j.compbiomed.2017.03.024.

[79] Fei-Fei, L.—Fergus, R.—Perona, P.: One-Shot Learning of Object Categories. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, 2006, No. 4, pp. 594–611, doi: 10.1109/TPAMI.2006.79.

[80] Koch, G.—Zemel, R.—Salakhutdinov, R.: Siamese Neural Networks for One-Shot Image Recognition. Proceedings of the 32nd International Conference on Machine Learning, Lille, France, JMLR: W & CP, Vol. 37, 2015, pp. 1–8.

[81] Snell, J.—Swersky, K.—Zemel, R.: Prototypical Networks for Few-Shot Learning. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 4077–4087.

[82] Triantafillou, E.—Zemel, R.—Urtasun, R.: Few-Shot Learning Through an Information Retrieval Lens. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 2255–2265.

[83] Wang, Y.—Yao, Q.: Few-Shot Learning: A Survey. arXiv:1904.05046v1, 2019.

[84] Wang, Y.—Yao, Q.—Kwok, J.—Ni, L. M.: Generalizing from a Few Examples: A Survey on Few-Shot Learning. arXiv:1904.05046v2, 2019.

[85] Bertinetto, L.—Valmadre, J.—Henriques, J. F.—Vedaldi, A.—Torr, P. H. S.: Fully-Convolutional Siamese Networks for Object Tracking. In: Hua, G., Jégou, H. (Eds.): Computer Vision – ECCV 2016 Workshops. Springer, Cham, Lecture Notes in Computer Science, Vol. 9914, 2016, pp. 850–865, doi: 10.1007/978-3-319-48881-3_56.

[86] Leal-Taixé, L.—Canton-Ferrer, C.—Schindler, K.: Learning by Tracking: Siamese CNN for Robust Target Association. 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 418–425, doi: 10.1109/cvprw.2016.59.

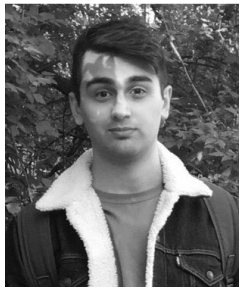
[87] Barthe, F.—Guédon, O.—Mendelson, S.—Naor, A.: A Probabilistic Approach to the Geometry of the ℓ_p^n-Ball. The Annals of Probability, Vol. 33, 2005, No. 2, pp. 480–513, doi: 10.1214/009117904000000874.

[88] Harman, R.—Lacko, V.: On Decompositional Algorithms for Uniform Sampling from n-Spheres and n-Balls. Journal of Multivariate Analysis, Vol. 101, 2010, pp. 2297–2304, doi: 10.1016/j.jmva.2010.06.002.

[89] LeCun, Y.—Bottou, L.—Bengio, Y.—Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, Vol. 86, 1998, No. 11, pp. 2278–2324, doi: 10.1109/5.726791.

Lev Utkin is Head of the Institute of Computer Science and Technology in Peter the Great Saint Petersburg Polytechnic University, Saint Petersburg, Russia. He is Professor and the Head of the Research Laboratory of Neural Network Technologies and Artificial Intelligence in the same university. In 1986 he graduated from the Saint Petersburg State Electrotechnical University (former Leningrad Electrotechnical Institute). He holds Ph.D. in information processing and control systems (1989) from the same university and D.Sc. in mathematical modelling (2001) from the Saint Petersburg State Institute of Technology, Russia. He was awarded an Alexander von Humboldt Foundation Fellowship (2001–2003). He is a member of the Society for Imprecise Probability Theory and Applications (SIPTA) and the International Society on Multiple Criteria Decision Making (ISMCDM). He is author of more than 300 scientific publications, including AI journals: Neurocomputing, Neural Networks, Knowledge-Based Systems, Applied Soft Computing, AI in Medicine, etc. His research interests are focused on machine learning, imprecise probability theory, decision making.

Maxim Kovalev is Ph.D. student at the Institute of Applied Mathematics and Mechanics in Peter the Great Saint Petersburg Polytechnic University, Saint Petersburg, Russia. He is Research Assistant at the Neural Network Technologies and Artificial Intelligence Laboratory in the same university. In 2019, he graduated from Peter the Great Saint Petersburg Polytechnic University, holding M.Sc. in bioinformatics. His research interests are focused on machine learning, explainable artificial intelligence, computational biology.



Ernest Kasimov is Research Assistant at the Neural Network Technologies and Artificial Intelligence Laboratory in Peter the Great Saint Petersburg Polytechnic University. In 2020, he graduated from the same university, holding M.Sc. in mathematics and computer science. His research interests are focused on machine learning.