

Discriminative Metric Learning with Deep Forest

Lev V. Utkin¹ and Mikhail A. Ryabinin²

Department of Telematics

Peter the Great St.Petersburg Polytechnic University

St.Petersburg, Russia

e-mail: ¹lev.utkin@gmail.com, ²mihail-ryabinin@yandex.ru

Abstract

A Discriminative Deep Forest (DisDF) as a metric learning algorithm is proposed in the paper. It is based on the Deep Forest or gcForest proposed by Zhou and Feng and can be viewed as a gcForest modification. The case of the fully supervised learning is studied when the class labels of individual training examples are known. The main idea underlying the algorithm is to assign weights to decision trees in random forest in order to reduce distances between objects from the same class and to increase them between objects from different classes. The weights are training parameters. A specific objective function which combines Euclidean and Manhattan distances and simplifies the optimization problem for training the DisDF is proposed. The numerical experiments illustrate the proposed distance metric algorithm.

Keywords: classification, random forest, decision tree, deep learning, metric learning, quadratic programming

1 Introduction

Real-world data usually has a high-dimensionality and a non-linear complex structure. In order to improve the performance of many classification algorithms especially those relying on distance computations (nearest neighbor classifiers, support vector machines, etc.) the distance metric learning methods are widely applied. Bellet et al. [1] selected three main groups of distance metric learning methods, which are defined by conditions of the available information about class labels of training examples. The first group consists of fully supervised algorithms for which we have a set of labeled training examples $S = \{(\mathbf{x}_i, y_i), i = 1, \dots, n\}$ such that every example has a label of a class $y_i \in \mathcal{Y}$, i.e., the class labels of individual training examples are known. The second group consists of weakly supervised algorithms. This group is characterized by the lack of class labels for training every example, but there is a side information in the form of constraints corresponding to the semantic similarity or dissimilarity of pairs of training examples. This can be seen as having label information only at the pair level. The third group consists of semi-supervised algorithms for which there is a part of data that are labeled or belong to similarity constraints, but another part consists of fully unlabeled data.

We study the first group of algorithms in this paper. The main goal of fully supervised distance metric learning is to use discriminative information to keep all the data samples in the same class close and those from different classes separated [18]. As indicated by Yang and Jin [28], unlike most supervised learning algorithms where training examples are given class labels, the training examples

of supervised distance metric learning is cast into pairwise constraints: the equivalence constraints where pairs of data points that belong to the same classes, and inequivalence constraints where pairs of data points belong to different classes.

Metric learning approaches were reviewed in [1, 5, 14, 29]. The basic idea underlying the metric learning solution is that the distance between similar objects should be smaller than the distance between different objects. If we have two observation vectors $\mathbf{x}_i \in \mathbb{R}^m$ and $\mathbf{x}_j \in \mathbb{R}^m$ from a training set, and the similarity of objects is defined by their belonging to the same class, then the distance $d(\mathbf{x}_i, \mathbf{x}_j)$ between the vectors should be minimized if \mathbf{x}_i and \mathbf{x}_j belong to the same class, and it should be maximized if \mathbf{x}_i and \mathbf{x}_j are from different classes. Several review papers analyze various methods and algorithms of metric learning [12, 19, 27]. A powerful implementation of the metric learning dealing with non-linear data structures is the so-called Siamese neural network introduced by Bromley et al. [4] in order to solve signature verification as a problem of image matching. This network consists of two identical sub-networks joined at their outputs. The two sub-networks extract features from two input examples during training, while the joining neuron measures the distance between the two feature vectors. The Siamese architecture has been exploited in many applications, for example, in face verification [7], in the one-shot learning in which predictions are made given only a single example of each new class [13], in constructing an inertial gesture classification [2], in deep learning [24], in extracting speaker-specific information [6], for face verification in the wild [11]. This is only a part of successful applications of Siamese neural networks. Many modifications of Siamese networks have been developed, including fully-convolutional Siamese networks [3], Siamese networks combined with a gradient boosting classifier [15], Siamese networks with the triangular similarity metric [29].

A new powerful method, which can be viewed as an alternative to deep neural networks, is the deep forest proposed by Zhou and Feng [30] and called the gcForest. It can be compared with a multi-layer neural network structure, but each layer in the gcForest contains many random forests instead of neurons. The gcForest can be regarded as an multi-layer ensemble of decision tree ensembles. Zhou and Feng [30] point out that their approach is highly competitive to deep neural networks. In contrast to deep neural networks which require great effort in hyperparameter tuning and large-scale training data, gcForest is much easier to train and can perfectly work when there are only small-scale training data. The deep forest solves the tasks of classification and regression. Therefore, by taking into account its advantages, it is important to modify it in order to develop a structure solving the metric learning task. We propose the so-called Discriminative Deep Forest (DisDF) which is a discriminative distance metric learning algorithm. It is based on the gcForest proposed by Zhou and Feng [30] and can be viewed as its modification. The main idea underlying the proposed DisDF is to assign weights to decision trees in random forest in order reduce distances between pairs of examples from the same class and to increase them between pairs of examples from different classes. We define the class distributions in the deep forest as the weighted sum of the tree class probabilities where the weights are determined by solving an optimization problem with the contrastive loss function as an objective function. The weights are viewed as training parameters. We also apply the greedy algorithm for training the DisDF, i.e., the weights are successively computed for every layer or level of the forest cascade. In order to efficiently find optimal weights, we propose to modify the standard contrastive loss in a way that makes the loss function to be convex with respect to the weights. This modification is carried out by combining Euclidean and Manhattan distances in the loss function. Moreover, we reduce the optimization problem for computing the weights to the standard convex quadratic optimization problem with linear constraints whose solution does not meet difficulties. For large-scale data, we apply the well-known Frank-Wolfe algorithm [9] which is very simple when the feasible set of weights is the unit simplex.

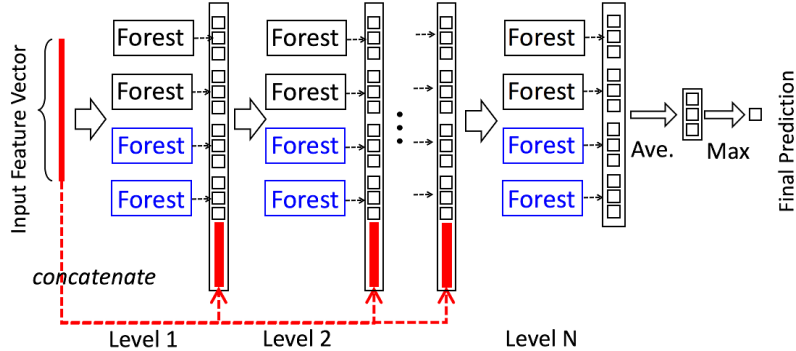


Figure 1: The architecture of the cascade forest [30]

The paper can be viewed as an extension of the results obtained by Utkin and Ryabinin [22] where the Siamese Deep Forest has been proposed. The main difference of the presented paper from [22] is that the case of the weakly supervised learning was used in the Siamese Deep Forest when there are no information about the class labels of individual training examples, but only information in the form of sets of semantically similar pairs is available. Now we study the case of the fully supervised learning when the class labels of individual training examples are known.

The paper is organized as follows. Section 2 gives a very short introduction into the gcForest proposed by Zhou and Feng [30]. The idea to assign weights to trees in random forests, which allows us to construct the DisDF, is considered in Section 3 in detail. Algorithms for training and testing the DisDF are considered in Section 4. Section 5 provides algorithms for dealing with large-scale training data. Numerical experiments with real data illustrating the proposed DisDF are given in Section 6. Concluding remarks are provided in Section 7.

2 Deep Forest

According to [30], the gcForest generates a deep forest ensemble, with a cascade structure. Representation learning in deep neural networks mostly relies on the layer-by-layer processing of raw features. The gcForest representational learning ability can be further enhanced by the so-called multi-grained scanning. Each level of cascade structure receives feature information processed by its preceding level, and outputs its processing result to the next level. Moreover, each cascade level is an ensemble of decision tree forests. We do not consider in detail the Multi-Grained Scanning where sliding windows are used to scan the raw features because this part of the deep forest is the same in the DisDF. However, the most interesting component of the gcForest from the DisDF construction point of view is the cascade forest.

Given an instance, each forest produces an estimate of class distribution by counting the percentage of different classes of examples at the leaf node where the concerned instance falls into, and then averaging across all trees in the same forest. The class distribution forms a class vector, which is then concatenated with the original vector to be input to the next level of cascade. The usage of the class vector as a result of the random forest classification is very similar to the idea underlying the stacking method [25]. The stacking algorithm trains the first-level learners using the

original training data set. Then it generates a new data set for training the second-level learner (meta-learner) such that the outputs of the first-level learners are regarded as input features for the second-level learner while the original labels are still regarded as labels of the new training data. In fact, the class vectors in the gcForest can be viewed as the meta-learners. In contrast to the stacking algorithm, the gcForest simultaneously uses the original vector and the class vectors (meta-learners) at the next level of cascade by means of their concatenation. This implies that the feature vector is enlarged and enlarged after every cascade level. The architecture of the cascade proposed by Zhou and Feng [30] is shown in Fig. 1. It can be seen from the figure that each level of the cascade consists of two different pairs of random forests which generate 3-dimensional class vectors concatenated each other and with the original input. After the last level, we have the feature representation of the input feature vector, which can be classified in order to get the final prediction. Zhou and Feng [30] propose to use different forests at every level in order to provide the diversity which is an important requirement for the random forest construction.

3 Weighted averages in forests

The DisDF aims to provide large distances between pairs of vectors belonging to the same class and small distances between vectors from different classes.

In order to achieve the above aim, we modify ideas provided by Xiong et al. [26] and Dong et al. [8]. Xiong et al. [26] considered an algorithm for solving the metric learning problem by means of the random forests. The proposed metric is able to implicitly adapt its distance function throughout the feature space. Dong et al. [8] proposed a random forest metric learning algorithm which combines semi-multiple metrics with random forests to better separate the desired targets and background in detecting and identifying target pixels based on specific spectral signatures in hyperspectral image processing. A common idea underlying the metric learning algorithms in [8] and [26] is to define a distance measure between a pair of training elements \mathbf{x}_i and \mathbf{x}_j for a combination of trees as average of some special functions of the training elements. For example, if a random forest is a combination of T decision trees $\{f_t(\mathbf{x}), t = 1, \dots, T\}$, then the distance measure is

$$d(\mathbf{x}_i, \mathbf{x}_j) = T^{-1} \sum_{t=1}^T f_t(\psi(\mathbf{x}_i, \mathbf{x}_j)). \quad (1)$$

Here $\psi(\mathbf{x}_i, \mathbf{x}_j)$ is a function specifically defined in [8] and [26].

The idea of the distance measure (1) produced by a random forest combined with the idea of probability distributions of classes for producing new augmented feature vectors after every level of the cascade forest proposed by Zhou and Feng [30] can be a basis for the modification of the gcForest which produce a new feature representation for efficient metric learning. According to [30], each forest of a cascade level produces an estimate of the class probability distribution by counting the percentage of different classes of training examples at a leaf node where the concerned instance falls into, and then averaging across all trees in the same forest. In contrast to this approach for computing the class probability distribution, we propose to define the class distribution as a weighted sum of the tree class probabilities. The weights can be viewed as training parameters. They are optimized in order to reduce distances between examples of the same class and to increase them between examples from different classes.

We apply the greedy algorithm for training the DisDF, i.e., we train separately every level starting from the first level such that every next level uses results of training obtained at the previous level.

Table 1: Notations for indices	
type	index
cascade level	$q = 1, \dots, Q$
forest	$k = 1, \dots, M_q$
tree	$t = 1, \dots, T_{k,q}$
class	$c = 1, \dots, C$

Let us introduce notations for indices corresponding to different deep forest components. The indices and their sets of values are shown in Table 1. One can see from Table 1, that there are Q levels of the deep forest, every level contains M_q forests such that every forest consists of $T_{k,q}$ trees. It is supposed that all training examples are divided into C classes.

Suppose we have trained trees in the DisDF. According to [30], the class distribution forms a class vector which is then concatenated with the original vector to be input to the next level of cascade. Suppose an origin vector is \mathbf{x}_i , and the $p_{i,c}^{(t,k,q)}$ is the probability of class c for \mathbf{x}_i produced by the t -th tree from the k -th forest at the cascade level q . Below we use the triple index (t, k, q) in order to indicate that the element belongs to the t -th tree from the k -th forest at the cascade level q . Following the results given in [30], the element $v_{i,c}^{(k,q)}$ of the class vector corresponding to class c and produced by the k -th forest in the gcForest is determined as

$$v_{i,c}^{(k,q)} = T_{k,q}^{-1} \sum_{t=1}^{T_{k,q}} p_{i,c}^{(t,k,q)}. \quad (2)$$

Denote the obtained class vector as $\mathbf{v}_i^{(k,q)} = (v_{i,1}^{(k,q)}, \dots, v_{i,C}^{(k,q)})$ and the concatenated M_q vectors $\mathbf{v}_i^{(k,q)}$ as $\mathbf{v}_i^{(q)}$. Then the concatenated vector $\mathbf{x}_i^{(1)}$ after the first level of the cascade is

$$\mathbf{x}_i^{(1)} = (\mathbf{x}_i, \mathbf{v}_i^{(1,1)}, \dots, \mathbf{v}_i^{(M_1,1)}) = (\mathbf{x}_i, \mathbf{v}_i^{(k,1)}, k = 1, \dots, M_1) = (\mathbf{x}_i, \mathbf{v}_i^{(1)}).$$

It is composed of the original vector \mathbf{x}_i and M_1 class vectors obtained from M_1 forests at the first level. In the same way, we can write the concatenated vector $\mathbf{x}_i^{(q)}$ after the q -th level of the cascade as

$$\begin{aligned} \mathbf{x}_i^{(q)} &= (\mathbf{x}_i^{(q-1)}, \mathbf{v}_i^{(1,q)}, \dots, \mathbf{v}_i^{(M_q,q)}) \\ &= (\mathbf{x}_i^{(q-1)}, \mathbf{v}_i^{(k,q)}, k = 1, \dots, M_q) = (\mathbf{x}_i^{(q-1)}, \mathbf{v}_i^{(q)}). \end{aligned} \quad (3)$$

Below we omit the index q in order to reduce the number of indices because all derivations will concern only level q , where q may be arbitrary from 1 to Q .

The vector \mathbf{x}_i in (3) is derived in accordance with the gcForest algorithm [30] by using (2). We propose to change the method for computing elements $v_{i,c}^{(k)}$ of the class vector in the DisDF, namely, the averaging (2) is replaced with the weighted sum of the form:

$$v_{i,c}^{(k)} = \sum_{t=1}^{T_k} p_{i,c}^{(t,k)} w^{(t,k)}. \quad (4)$$

Here $w^{(t,k)}$ is a weight for combining the class probabilities of the t -th tree from the k -th forest. An illustration of the weighted averaging is shown in Fig. 2, where we partly modify a picture from

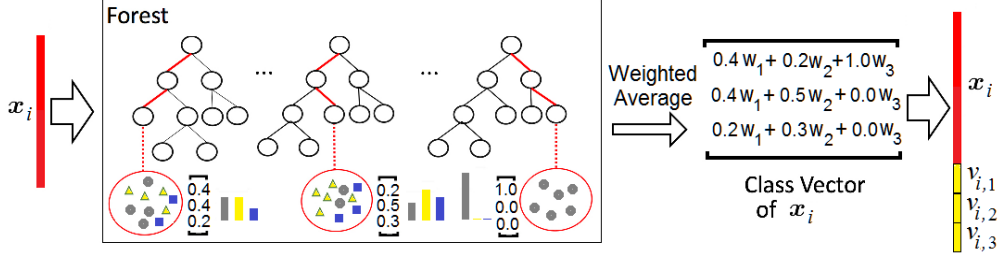


Figure 2: An illustration of the class vector generation taking into account the weights

[30] (the left part is copied from [30, Fig. 2]) in order to show how elements of the class vector are derived as a simple weighted sum. One can see from Fig. 2 that three-class distribution is estimated by counting the percentage of different classes of a new training example \mathbf{x}_i at the leaf node where the concerned example \mathbf{x}_i falls into. Then the class vector of \mathbf{x}_i is computed as the weighted average. It is important to note that we weigh trees belonging to one of the forests, but we do not weigh classes, i.e., the weights do not depend on the class c . Moreover, the weights characterize trees, but not training elements. One can also see from Fig. 2 that the augmented features $v_{i,c}^{(k)}$, $c = 1, \dots, C$, corresponding to the q -th forest are obtained as weighted sums, i.e., there hold

$$\begin{aligned} v_{i,1}^{(k)} &= 0.4w^{(1,k)} + 0.2w^{(2,k)} + 1.0w^{(3,k)}, \\ v_{i,2}^{(k)} &= 0.4w^{(1,k)} + 0.5w^{(2,k)} + 0.0w^{(3,k)}, \\ v_{i,3}^{(k)} &= 0.2w^{(1,k)} + 0.3w^{(2,k)} + 0.0w^{(3,k)}. \end{aligned}$$

The weights are restricted by the following obvious condition:

$$\sum_{t=1}^{T_k} w^{(t,k)} = 1, \quad w^{(t,k)} \geq 0, \quad t = 1, \dots, T_k. \quad (5)$$

So, we have the weighted averages for every forest, and the weights are trained parameters which are optimized in order to decrease the distance between objects from the same class and to increase the distance between objects from different classes. Therefore, the next task is to develop an algorithm for training the DisDF, in particular, for computing the weights for every forest and for every cascade level.

4 The DisDF training and testing

In this section, we consider how to efficiently compute the weights in the DisDF. The main difficulty in solving this task is to choose a proper loss function which allows us to implement efficient computations. We overcome the difficulty by modifying the well-known contrastive loss function. The proposed modification gives us a convex loss function with respect to weights.

Before considering the DisDF training, we introduce the following notation:

$$\begin{aligned}
P_{ij}^{(t,k)} &= \sum_{c=1}^C \left(p_{i,c}^{(t,k)} - p_{j,c}^{(t,k)} \right)^2, \\
Q_{ij}^{(t,k)} &= \sum_{c=1}^C \left| p_{i,c}^{(t,k)} - p_{j,c}^{(t,k)} \right|, \\
\pi^{(k,t)} &= \sum_{i,j} (1 - z_{ij}) P_{ij}^{(t,k)}, \\
\mathbf{P}_{ij}^{(k)} &= \left(P_{ij}^{(t,k)}, t = 1, \dots, T_k \right), \quad \mathbf{P}_{ij} = \left(\mathbf{P}_{ij}^{(k)}, k = 1, \dots, M \right), \\
\mathbf{Q}_{ij}^{(k)} &= \left(Q_{ij}^{(t,k)}, t = 1, \dots, T_k \right), \quad \mathbf{Q}_{ij} = \left(\mathbf{Q}_{ij}^{(k)}, k = 1, \dots, M \right), \\
\mathbf{w}^{(k)} &= \left(w^{(t,k)}, t = 1, \dots, T_k \right), \quad \mathbf{w} = \left(\mathbf{w}^{(k)}, k = 1, \dots, M \right), \\
\left(\mathbf{w}^{(k)} \right)^2 &= \left(\left(w^{(t,k)} \right)^2, t = 1, \dots, T_k \right), \quad \mathbf{w}^2 = \left(\left(\mathbf{w}^{(k)} \right)^2, k = 1, \dots, M \right), \\
\pi^{(k)} &= \left(\pi^{(t,k)}, t = 1, \dots, T_k \right), \quad \pi = \left(\pi^{(k)}, k = 1, \dots, M \right).
\end{aligned}$$

Vectors \mathbf{P}_{ij} , \mathbf{Q}_{ij} , \mathbf{w} , \mathbf{w}^2 , π have the same length $\sum_{k=1}^M T_k$, and they are produced as the concatenation of M vectors characterizing the forests, for example, the vector \mathbf{w} is the concatenation of vectors $\mathbf{w}^{(k)} = (w^{(1,k)}, \dots, w^{(T_k,k)})$, $k = 1, \dots, M$. If a pair of examples $(\mathbf{x}_i, \mathbf{x}_j)$ belongs to the same class, then we assume that $z_{ij} = 0$, otherwise $z_{ij} = 1$.

We apply the greedy algorithm for training the DisDF, namely, we train separately every level starting from the first level such that every next level uses results of training at the previous level. The training process at every level consists of two steps. The first step aims to train all trees by applying all training examples. This step totally coincides with the training algorithm of the original gcForest proposed by Zhou and Feng [30].

The second step is to train the DisDF in order to get the weights $w^{(t,k)}$, $t = 1, \dots, T_k$. This can be done by minimizing the following objective function over M unit (probability) simplices in \mathbb{R}^{T_k} denoted as Δ_k , i.e., over non-negative vectors $\mathbf{w}^{(k)}$, $k = 1, \dots, M$, that sum up to one:

$$\min_{\mathbf{w}} J_q(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i,j} l(\mathbf{x}_i, \mathbf{x}_j, y_i, y_j, \mathbf{w}) + \lambda R(\mathbf{w}). \quad (6)$$

Here l is the loss function, $R(\mathbf{w})$ is a regularization term, λ is a hyper-parameter which controls the strength of the regularization. We define the regularization term as

$$R(\mathbf{w}) = \|\mathbf{w}\|^2.$$

One of the most popular loss functions in metric learning is the contrastive loss which can be written in terms of the considered problem as follows:

$$l(\mathbf{x}_i, \mathbf{x}_j, y_i, y_j, \mathbf{w}) = ((1 - z_{ij})d(\mathbf{x}_i, \mathbf{x}_j) + z_{ij} \max(0, \tau - d(\mathbf{x}_i, \mathbf{x}_j))). \quad (7)$$

Here τ is the tuning parameter. It can be seen from the above expression that the first term of the sum corresponds to the reduction of distances between points of the same class, and the second

term increases the distances between points from different classes. We aim to find the minimum of the function with respect to \mathbf{w} in order to find an optimal vector of weights which fulfills the above properties of examples from the same and from different classes. Moreover, we aim to get the convex function $J_q(\mathbf{w})$ with respect to \mathbf{w} satisfying (5) in order to simplify the solution of the optimization problem.

Since the weights \mathbf{w} impact on values of augmented parts of vectors \mathbf{x}_i , then we define the distances between these parts of vectors without taking into account the concatenated original vectors. Let us denote the distance between two vectors $\mathbf{v}_i^{(k)}$ and $\mathbf{v}_j^{(k)}$ as $d(\mathbf{v}_i^{(k)}, \mathbf{v}_j^{(k)})$. Then the k -th forest at level q produces the following Euclidean distance:

$$\begin{aligned} d(\mathbf{v}_i^{(k)}, \mathbf{v}_j^{(k)}) &= \sum_{c=1}^C (v_{i,c}(k) - v_{j,c}(k))^2 \\ &= \sum_{c=1}^C \sum_{t=1}^{T_k} (p_{i,c}^{(t,k)} - p_{j,c}^{(t,k)})^2 (w^{(t,k)})^2 \\ &= \sum_{t=1}^{T_k} P_{ij}^{(t,k)} (w^{(t,k)})^2. \end{aligned}$$

Taking into account all M forests at level q , we can write the total distance between \mathbf{v}_i and \mathbf{v}_j without original vectors as:

$$d(\mathbf{v}_i, \mathbf{v}_j) = \sum_{k=1}^M \sum_{t=1}^{T_k} P_{ij}^{(t,k)} (w^{(t,k)})^2 = \langle \mathbf{P}_{ij}, \mathbf{w}^2 \rangle. \quad (8)$$

Here $\langle \cdot, \cdot \rangle$ is the dot product of two vectors. The function (8) is convex with respect to $w^{(t,k)}$. Unfortunately, the second term in (7) as a function of \mathbf{w} is non-convex. In order to overcome this difficulty, we modify the contrastive loss and reformulate the distance metric in the second term as follows:

$$d_1(\mathbf{v}_i, \mathbf{v}_j) = \|\mathbf{v}_i - \mathbf{v}_j\|_1 = \sum_{k=1}^M \sum_{t=1}^{T_k} Q_{ij}^{(t,k)} w^{(t,k)} = \langle \mathbf{Q}_{ij}, \mathbf{w} \rangle.$$

The Manhattan distance is used instead of the Euclidean distance. We do not need to consider the absolute values of $w^{(t,k)}$ because the weights are restricted by condition (5) such that their values are non-negative. Moreover, we rewrite the second term in the following form:

$$z_{ij} (\max(0, \tau - d_1(\mathbf{v}_i, \mathbf{v}_j)))^2.$$

The above function is convex in the interval $[0, 1]$ of $w^{(t,k)}$. Then the objective function $J_q(\mathbf{w})$ as a sum of the convex functions is convex with respect to weights. Finally, we write the function $J_q(\mathbf{w})$ as

$$\begin{aligned} J_q(\mathbf{w}) &= \min_{\mathbf{w}} \sum_{i,j} (1 - z_{ij}) \langle \mathbf{P}_{ij}, \mathbf{w}^2 \rangle \\ &\quad + \sum_{i,j} z_{ij} (\max(0, \tau - \langle \mathbf{Q}_{ij}, \mathbf{w} \rangle))^2 + \lambda \|\mathbf{w}\|^2. \end{aligned} \quad (9)$$

Let us introduce new variables

$$\alpha_{ij} = \max(0, \tau - \langle \mathbf{Q}_{ij}, \mathbf{w} \rangle).$$

Then we can rewrite the optimization problem as

$$J_q(\mathbf{w}) = \min_{\alpha_{ij}, \mathbf{w}} \left(\langle \pi, \mathbf{w}^2 \rangle + \sum_{i,j} z_{ij} \alpha_{ij}^2 + \lambda \|\mathbf{w}\|^2 \right), \quad (10)$$

subject to (5) and

$$\alpha_{ij} \geq \tau - \langle \mathbf{Q}_{ij}, \mathbf{w} \rangle, \quad \alpha_{ij} \geq 0, \quad \forall i, j. \quad (11)$$

This is a standard quadratic optimization problem with linear constraints and with $M \cdot T_k$ variables $w^{(t,k)}$ and $n \cdot (n-1)/2$ variables α_{ij} .

Let us prove that the problem can be decomposed into M quadratic optimization problems in accordance with forests, i.e., every optimization problem can be solved for every forest separately. Let us return to the problem (9). The first term in the objective function can be rewritten as

$$\sum_{i,j} (1 - z_{ij}) \langle \mathbf{P}_{ij}, \mathbf{w}^2 \rangle = \sum_{k=1}^M \left[\sum_{i,j} (1 - z_{ij}) \sum_{t=1}^{T_k} P_{ij}^{(t,k)} \left(w^{(t,k)} \right)^2 \right].$$

Since constraints (5) for $w^{(t,k_0)}$ and $w^{(t,k_1)}$ do not intersect each other by $k_0 \neq k_1$, i.e., they consist of different weights, then we can separately consider M optimization problems for every k .

Let us consider the second term in (9) and rewrite it as follows:

$$\begin{aligned} & \sum_{i,j} z_{ij} (\max(0, \tau - \langle \mathbf{Q}_{ij}, \mathbf{w} \rangle))^2 \\ &= \sum_{i,j} z_{ij} \left(\max \left(0, \sum_{k=1}^M \left[\tau_k - \sum_{t=1}^{T_k} Q_{ij}^{(t,k)} \left(w^{(t,k)} \right) \right] \right) \right)^2. \end{aligned}$$

Here $\sum_{k=1}^M \tau_k = \tau$. In order to minimize $J_q(\mathbf{w})$, we need to maximize $\tau_k - \sum_{t=1}^{T_k} Q_{ij}^{(t,k)} \left(w^{(t,k)} \right)$. But its maximizing does not depend on k because the corresponding constraints (5) for $w^{(t,k_0)}$ and $w^{(t,k_1)}$ do not intersect each other. Since τ as well as τ_k are tuning parameters, then we can tune τ_k for every $k = 1, \dots, M$, separately. Then we can write

$$\sum_{i,j} z_{ij} \left(\max \left(0, \left[\tau_{k_0} + A - \sum_{t=1}^{T_k} Q_{ij}^{(t,k)} w^{(t,k)} \right] \right) \right)^2.$$

Here A is a term which does not depend on $w^{(t,k)}$, but, of course, it depends on other weights from the vector $\mathbf{w}^{(k)}$. In fact, we have the parameter $\tau_k = \tau_{k_0} + A$ for tuning. So, we can separately solve the problems for every forest. Then (10)-(11) can be rewritten for every forest as follows:

$$J_q(\mathbf{w}^{(k)}) = \min_{\alpha_{ij}, \mathbf{w}^{(k)}} \left(\left\langle \pi^{(k)}, \left(\mathbf{w}^{(k)} \right)^2 \right\rangle + \sum_{i,j} z_{ij} \alpha_{ij}^2 + \lambda \left\| \mathbf{w}^{(k)} \right\|^2 \right), \quad (12)$$

subject to (5) and

$$\alpha_{ij} \geq \tau - \langle \mathbf{Q}_{ij}^{(k)}, \mathbf{w}^{(k)} \rangle, \quad \alpha_{ij} \geq 0, \quad \forall i, j. \quad (13)$$

In sum, we can write a general algorithm for training the DisDF (see Algorithm 1). Its complexity mainly depends on the number of levels. Having the trained DisDF, we can classify a new example \mathbf{x} . By using the trained decision trees and the weights \mathbf{w} , the vector \mathbf{x} is augmented at each level. Finally, we get the vector \mathbf{v} of augmented features after the Q -th level of the forest cascade corresponding to original example \mathbf{x} .

The class of the example \mathbf{x} is defined by the sum of the c -th elements of vectors $\mathbf{v}^{(1,Q)}, \dots, \mathbf{v}^{(M_Q,Q)}$. The example \mathbf{x} belongs to the class c , if the sum of the c -th elements $v_c^{(1,Q)} + \dots + v_c^{(M_Q,Q)}$ is maximal.

Algorithm 1 A general algorithm for training the DDF

Require: Training set $S = \{(\mathbf{x}_i, y_i), i = 1, \dots, n\}$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{1, \dots, C\}$; number of levels Q

Ensure: \mathbf{w} for every $q = 1, \dots, Q$

- 1: **for** $q = 1, q \leq Q$ **do**
 - 2: Train all trees at the q -th level in accordance with the gcForest algorithm [30]
 - 3: **for** $k = 1, k \leq M_q$ **do**
 - 4: Compute the weights $\mathbf{w}^{(k)}$ by solving the k -th quadratic optimization problem with the objective function (12) and constraints (5) and (13)
 - 5: **end for**
 - 6: Concatenate $\mathbf{w}^{(k)}, k = 1, \dots, M_q$, to get \mathbf{w}
 - 7: For every \mathbf{x}_i , compute $\mathbf{v}_i^{(k)}$ by using (4), $k = 1, \dots, M_q$, and \mathbf{v}_i
 - 8: For every \mathbf{x}_i , form the concatenated vector $\mathbf{x}_i^{(q)}$ for the next level by using (3)
 - 9: **end for**
-

5 Large-scale data

The main difficulty in solving the problem (12)-(13) is that the number of variables as well as the number of constraints may be extremely large because we need to enumerate all pairs of training data. Therefore, we simplify the algorithm by using the well-known Frank-Wolfe algorithm [9]. It is represented as Algorithm 2.

Denote

$$S_{ij} = \tau - \langle \mathbf{Q}_{ij}^{(k)}, \mathbf{w}^{(k)} \rangle.$$

The gradient of the function $J_q(\mathbf{w}^{(k)})$ with respect to the variable $w^{(k,t)}$ is

$$\begin{aligned} \nabla_{w^{(k,t)}} J_q(\mathbf{w}^{(k)}) &= 2w^{(k,t)} \left[\lambda + \sum_{i,j} (1 - z_{ij}) P_{ij}^{(k,t)} \right] \\ &\quad - 2 \sum_{i,j} z_{ij} \cdot \begin{cases} 0, & S_{ij} \leq 0, \\ S_{ij}^2 \cdot Q_{ij}^{(k,t)}, & S_{ij} > 0. \end{cases} \end{aligned} \quad (14)$$

It should be noted that the linear problem in Algorithm 2 can be solved by looking for the solution among T_k vertices of the unit simplex Δ_k . The vertices are of the form:

$$\mathbf{g} = (0, \dots, 0, 1, 0, \dots, 0).$$

Algorithm 2 The Frank-Wolfe algorithm [9] in terms of the DDF training

Require: $\mathbf{P}_{ij}^{(k)}$, $\mathbf{Q}_{ij}^{(k)}$, z_{ij} , λ , τ ; number of iterations S

Ensure: $\mathbf{w}^{(k)}$

- 1: Initialize $\mathbf{w}_0 \in \Delta_k$, Δ_k is the unit simplex having T_k vertices
 - 2: **for** $s = 0, s \leq S - 1$ **do**
 - 3: Compute the gradient $f(\mathbf{w}_s) = \nabla_{\mathbf{w}^{(k)}} J_q(\mathbf{w}_s)$
 - 4: Solve the linear problem $\mathbf{g}_s \leftarrow \arg \min_{\mathbf{g} \in \Delta_k} \langle \mathbf{g}, f(\mathbf{w}_s) \rangle$, $\mathbf{g} \in \mathbb{R}^{T_k}$ is the vector of optimization variables
 - 5: Compute $\gamma_s \leftarrow 2/(s + 2)$
 - 6: Update $\mathbf{w}_{s+1} \leftarrow \mathbf{w}_s + \gamma_s (\mathbf{g}_s - \mathbf{w}_s)$
 - 7: **end for**
 - 8: $\mathbf{w}^{(k)} \leftarrow \mathbf{w}_{s+1}$
-

Hence, we have to look for smallest value of $f(\mathbf{w}_s)$ by $t = 1, \dots, T_k$. In other words, we compute $\nabla_{\mathbf{w}^{(k)}} J_q(\mathbf{w}^{(k)})$ for different t . Then \mathbf{g}_s consists of $T_k - 1$ zero elements and the unit element whose index t_0 coincides with the index of the smallest value of $f(\mathbf{w}_s)$.

So, we have obtained a very simple algorithm for solving the problem (12)-(13). Its simplicity is due to simple constraints for the weights $w^{(k,t)}$. The modified Frank-Wolfe algorithm is represented as Algorithm 3.

Algorithm 3 The modified Frank-Wolfe algorithm

Require: $\mathbf{P}_{ij}^{(k)}$, $\mathbf{Q}_{ij}^{(k)}$, z_{ij} , λ , τ ; number of iterations S

Ensure: $\mathbf{w}^{(k)}$

- 1: Initialize $\mathbf{w}_0 \in \Delta_k$, Δ_k is the unit simplex having T_k vertices
 - 2: **for** $s = 0, s \leq S - 1$ **do**
 - 3: Compute $\nabla_{w^{(k,t)}} J_q(\mathbf{w}^{(k)})$ for every $t = 1, \dots, T_k$ by using (14)
 - 4: Compute $t_0 \leftarrow \arg \min_{t=1, \dots, T_k} \nabla_{w^{(k,t)}} J_q(\mathbf{w}^{(k)})$
 - 5: $\mathbf{g}_s \leftarrow (0, \dots, 0, 1_{t_0}, 0, \dots, 0)$
 - 6: Compute $\gamma_s \leftarrow 2/(s + 2)$
 - 7: Update $\mathbf{w}_{s+1} \leftarrow \mathbf{w}_s + \gamma_s (\mathbf{g}_s - \mathbf{w}_s)$
 - 8: **end for**
 - 9: $\mathbf{w}^{(k)} \leftarrow \mathbf{w}_{s+1}$
-

6 Numerical experiments

We compare the DisDF with the gcForest. The DisDF has the same cascade structure as the standard gcForest described in [30]. Each level (layer) of the cascade structure consists of 2 complete-random tree forests and 2 random forests. Three-fold cross-validation is used for the class vector generation. The number of cascade levels is automatically determined.

We modify a software in Python implementing the gcForest and available at <https://github.com/leopiney/deep-forest> to implement the procedure for computing optimal weights and weighted averages $v_{i,c}^{(k)}$. Accuracy measure A used in numerical experiments is the proportion of correctly classified cases on a sample of data. To evaluate the average accuracy, we

Table 2: The DisDF accuracy for the Parkinsons data set by different N and T in every forest

T	100		400		700		1000	
N	gcF	DisDF	gcF	DisDF	gcF	DisDF	gcF	DisDF
50	0.80	0.84	0.85	0.87	0.84	0.84	0.79	0.80
80	0.815	0.87	0.875	0.90	0.85	0.875	0.80	0.85
100	0.833	0.86	0.88	0.92	0.86	0.88	0.83	0.90
120	0.84	0.883	0.92	0.95	0.92	0.95	0.92	0.95

perform a cross-validation with 100 repetitions, where in each run, we randomly select N training data and $N_{\text{test}} = 2N/3$ test data.

First, we compare the DisDF with the gcForest by using some public data sets from UCI Machine Learning Repository [17]: the Ecoli data set (336 instances, 8 features, 8 classes), the Parkinsons data set (197 instances, 23 features, 2 classes), the Ionosphere data set (351 instances, 34 features, 2 classes). A more detailed information about the data sets can be found from, respectively, the data resources. Different values for the regularization hyper-parameter λ have been tested, choosing those leading to the best results. In order to investigate how the number of decision trees impact on the classification accuracy, we study the DisDF as well as gcForest by different number of trees, namely, we take $T_k = T = 100, 400, 700, 1000$.

Results of numerical experiments for the Parkinsons data set are shown in Table 2. It contains the DisDF accuracy measures obtained for the gcForest (denoted as gcF) and the DisDF as functions of the number of trees T in every forest and the number $N = 50, 80, 100, 120$ of examples in the training set. It follows from Table 2 that the accuracy of the DisDF exceeds the same measure of the gcForest in most cases. The difference is not significant by $N = 50$ and 120. However, it is larger by $N = 100$ and the small amount of trees T . Nevertheless, the largest difference between accuracy measures of the DisDF and the gcForest is observed by $T = 1000$ and $N = 100$.

Results of numerical experiments for the Ecoli data set are shown in Table 3. This data set shows the largest difference between accuracy measures of the DisDF and gcForest by $N = 50$. This implies that the proposed DisDF outperforms the gcForest by the very small amount of training data. It is interesting to note that the DisDF does not outperform the gcForest by $N = 120$ and $T = 700$. At the same time, the number of trees also significantly impact on the accuracy, namely, we can see from Table 3 that the outperformance of the DisDF is observed by $T = 100$ and 400.

Numerical results for the Ionosphere data set are represented in Table 4. It follows from Table 4 that the largest difference between accuracy measures of the DisDF and the gcForest is observed by $T = 100$ and $N = 50$. This again implies that the DisDF outperforms the gcForest by the very small amount of training data.

By analyzing all results, we have to point out that, in contrast to comparative results, the highest accuracy measure can be obtained by the large number of training data and the large number of trees in every forest. If the first parameter (N) cannot be controlled, then the number of trees is a tuning parameter. It is interesting to note that the largest number of trees by some fixed values of N , for example, $N = 80$ and 100, does not provide the largest accuracy measure. In particular, we can see from Tables 3-4 that the largest accuracy measures are achieved at $T = 400$ and 700.

It should be noted that the multi-grained scanning proposed in [30] was not applied to investigating the above data sets having relatively small numbers of features. The above numerical results have been obtained by using only the forest cascade structure.

Another data set for comparison of the DisDF and gcForest is the well-known MNIST data set

Table 3: The DisDF accuracy for the Ecoli data set by different N and T in every forest

T	100		400		700		1000	
N	gcF	DisDF	gcF	DisDF	gcF	DisDF	gcF	DisDF
50	0.78	0.82	0.80	0.86	0.72	0.76	0.73	0.77
80	0.88	0.89	0.81	0.83	0.87	0.90	0.84	0.88
100	0.85	0.90	0.85	0.87	0.82	0.84	0.90	0.93
120	0.84	0.88	0.80	0.85	0.93	0.92	0.95	0.96

Table 4: The DisDF accuracy for the Ionsphere data set by different N and T in every forest

T	100		400		700		1000	
N	gcF	DisDF	gcF	DisDF	gcF	DisDF	gcF	DisDF
50	0.48	0.66	0.58	0.50	0.48	0.58	0.40	0.60
80	0.71	0.78	0.68	0.70	0.66	0.65	0.72	0.75
100	0.72	0.80	0.74	0.78	0.74	0.78	0.76	0.775
120	0.69	0.70	0.77	0.80	0.81	0.82	0.83	0.83

which is a commonly used large database of 28×28 pixel handwritten digit images [16]. It has a training set of 60,000 examples, and a test set of 10,000 examples. The digits are size-normalized and centered in a fixed-size image. The data set is available at <http://yann.lecun.com/exdb/mnist/>. In contrast to gcForest, we did not use the multi-grained scanning scheme for the DisDF implementation because its use provides worse results by the small amount of training data. Results of numerical experiments for the MNIST data set are shown in Table 5. Numerical experiments with the MNIST data set by the very small training data have shown that the use of the multi-grained scanning procedure may deteriorate the classification performance of the DisDF as well as the gcForest. Therefore, we did not use this procedure in numerical experiments with the MNIST. It can be seen from Table 5 that the DisDF outperforms the gcForest in the most cases. The largest difference between accuracy measures of the DisDF and the gcForest is observed by $T = 100$ and $N = 50$.

7 Conclusion

A discriminative metric learning algorithm in the form of the DisDF has been presented in the paper. Two main contributions should be pointed out. First, we have introduced weights for trees which allow us to apply some new properties for the deep forest. The weights play a key role in the developing the DisDF. This role is similar to the role of weights of connections in neural networks

Table 5: The DisDF accuracy for the MNIST data set by different N and T in every forest

T	100		400		700		1000	
N	gcF	DisDF	gcF	DisDF	gcF	DisDF	gcF	DisDF
50	0.63	0.70	0.67	0.71	0.68	0.68	0.69	0.70
80	0.63	0.70	0.75	0.77	0.75	0.77	0.70	0.70
100	0.70	0.73	0.76	0.78	0.76	0.78	0.76	0.78
120	0.74	0.75	0.76	0.76	0.76	0.78	0.77	0.78

which also have to be trained. This implies that we can control properties of the deep forest and its modifications by constructing the corresponding objective functions $J(\mathbf{w})$ which are similar to the loss or reconstruction functions in neural networks. This fact opens a way for developing new modifications of the deep forest which have certain properties. Moreover, we get an opportunity to consider the relationship between the deep forest and neural networks as it has been done by Richmond et al. [21] in exploring the relationship between stacked random forests and deep convolutional neural networks. Second, we have used a new objective function $J_q(\mathbf{w})$ which combines two different distance metrics: Euclidean and Manhattan distances. This combination allows us to get the convex objective function with respect to \mathbf{w} and to significantly simplify the optimization problem.

It should be also noted that one of the implementations of the DisDF has been represented in the paper. It should be noted that other modifications of the DisDF can be also obtained. We can use more efficient modifications of the Frank-Wolfe algorithm, for example, algorithms proposed by Hazan and Luo [10] or Reddi et al. [20]. We can also consider non-linear functions of weights like activation functions in neural networks. Moreover, we can investigate imprecise statistical models [23] for restricting the set of weights, for example, we can reduce the unit simplex of weights in order to get robust classification models. These modifications can be viewed as directions for further research.

Acknowledgement

The reported study was partially supported by RFBR, research project No. 17-01-00118.

References

- [1] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 28 Jun 2013.
- [2] S. Berlemont, G. Lefebvre, S. Duffner, and C. Garcia. Siamese neural network based similarity metric for inertial gesture classification and rejection. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 1, pages 1–6. IEEE, May 2015.
- [3] L. Bertinetto, J. Valmadre, J.F. Henriques, A. Vedaldi, and P.H.S. Torr. Fully-convolutional siamese networks for object tracking. *arXiv:1606.09549v2*, 14 Sep 2016.
- [4] J. Bromley, J.W. 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*, 7(4):737–744, 1993.
- [5] H. Le Capitaine. Constraint selection in metric learning. *arXiv:1612.04853v1*, 14 Dec 2016.
- [6] K. Chen and A. Salman. Extracting speaker-specific information with a regularized siamese deep network. In *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, pages 298–306. Curran Associates, Inc., 2011.
- [7] 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)*, volume 1, pages 539–546. IEEE, 2005.

- [8] Y. Dong, B. Du, and L. Zhang. Target detection based on random forest metric learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(4):1830–1838, 2015.
- [9] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, March 1956.
- [10] E. Hazan and H. Luo. Variance-reduced and projection-free stochastic optimization. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *ICML ’16*, pages 1263–1271, 2016.
- [11] J. Hu, J. Lu, and Y.-P. Tan. Discriminative deep metric learning for face verification in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1875–1882. IEEE, 2014.
- [12] D. Kedem, S. Tyree, K. Weinberger, F. Sha, and G. Lanckriet. Non-linear metric learning. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2582–2590. Curran Associates, Inc., 2012.
- [13] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1–8, Lille, France, 2015.
- [14] B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.
- [15] L. Leal-Taixe, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese cnn for robust target association. *arXiv preprint arXiv:1604.07866*, 26 Apr 2016.
- [16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [17] M. Lichman. UCI machine learning repository, 2013.
- [18] Y. Mu and W. Ding. Local discriminative distance metrics and their real world applications. In *2013 IEEE 13th International Conference on Data Mining Workshops (ICDMW)*, pages 1145–1152. IEEE, Dec 2013.
- [19] M. Norouzi, D. Fleet, and R. Salakhutdinov. Hamming distance metric learning. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1070–1078. Curran Associates, Inc., 2012.
- [20] S.J. Reddi, S. Sra, B. Póczos, and A. Smola. Stochastic frank-wolfe methods for nonconvex optimization. *arXiv:1607.08254v2*, July 2016.
- [21] D.L. Richmond, D. Kaimueller, M. Yang, E.W. Myers, and C. Rother. Mapping stacked decision forests to deep and sparse convolutional neural networks for semantic segmentation. *arXiv:1507.07583v2*, Dec 2015.
- [22] L.V. Utkin and M.A. Ryabinin. A Siamese deep forest. *arXiv:1704.08715v1*, Apr 2017.
- [23] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

- [24] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, C.K. Luk, and G. Wang. Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2016.
- [25] D.H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.
- [26] C. Xiong, D. Johnson, R. Xu, and J.J. Corso. Random forests for metric learning with implicit pairwise position dependence. arXiv:1201.0610v1, Jan 2012.
- [27] Z. Xu, K.Q. Weinberger, and O. Chapelle. Distance metric learning for kernel machines. arXiv:1208.3422, 2012.
- [28] L. Yang and R. Jin. Distance metric learning: a comprehensive survey. Technical report, Michigan State University, 2006.
- [29] L. Zheng, S. Duffner, K Idrissi, C. Garcia, and A. Baskurt. Siamese multi-layer perceptrons for dimensionality reduction and face identification. *Multimedia Tools and Applications*, 75(9):5055–5073, 2016.
- [30] Z.-H. Zhou and J. Feng. Deep forest: Towards an alternative to deep neural networks. arXiv:1702.08835v1, February 2017.