ELSEVIER

Pattern Recognition Letters
journal homepage: www.elsevier.com

Deterministic Dropout for Deep Neural Networks Using Composite Random Forest

Bikash Santra[a,**], Angshuman Paul[*,a], Dipti Prasad Mukherjee[a]

[a]*Electronics and Communication Sciences Unit, Indian Statistical Institute, 203, B T Road, Kolkata, 700108, India*

ABSTRACT

Dropout prevents overfitting in deep neural networks. Typical strategy of dropout involves random termination of connections irrespective of their importance. Termination blocks the propagation of class discriminative information across the network. As a result, dropout may lead to inferior performance. We propose a deterministic dropout where only unimportant connections are dropped ensuring propagation of class discriminative information. We identify the unimportant connections using a novel composite random forest, integrated into the network. We prove that better generalization is achieved by terminating these unimportant connections. The proposed algorithm is useful in preventing overfitting in noisy datasets. The proposal is equally good for datasets with smaller number of training examples. Experiments on several benchmark datasets show up to 8% improvement in classification accuracy.

Keywords: Deterministic dropout, Composite random forest, Deep neural network, Regularizer

1. Introduction

Dropout (Hinton et al., 2012; Srivastava et al., 2014) is arguably the most popular strategy (Cheng et al., 2019; Calvo-Zaragoza et al., 2019) to avoid overfitting in neural networks. However, there are several shortcomings of this technique. First, connections to neurons are dropped out randomly. As a result, some neurons that may have produced or propagated class discriminative attributes may be dropped out leading to loss of information (Zhao et al., 2019). Second, we have no means to control dropout based on the performance of the network during training. The role of a neuron-pair, connected by a weight, is not exploited to drop out the connection between neuron-pair. Therefore, dropout is not dependent on the complex interplay of neurons between a pair of network layers.

DropConnect (Wan et al., 2013) is a generalization of the dropout mechanism where a randomly selected subset of weights in the network is set to zero. However, the aforementioned problems of dropout are present in DropConnect as well.

Concrete dropout (Gal et al., 2017) is a variant of dropout that allows dropout probabilities to be tuned. Instead of performing dropout randomly, the above approach calibrates uncertainties that governs dropout. This helps the dropout probabilities to be tuned using gradient descent methods. Although this method yields superior performance over the classical dropout mechanism (Hinton et al., 2012; Srivastava et al., 2014), the aforementioned shortcomings of dropout can not be avoided in this method.

In (LeCun et al., 1990), the authors explored the dependence of generalization ability of a neural network on the saliency of connections in the network. We define a novel importance measure for a connection to estimate the saliency of the connection. The importance of a connection is evaluated by capturing the complex interplay of neurons through a novel composite random forest integrated inside the network. Subsequently, we propose a deterministic dropout where only unimportant connections of a network are terminated. Following (LeCun et al., 1990), we show that the removal of unimportant connections through the proposed deterministic dropout leads to better generalization. Thus our method helps to get rid of overfitting in a principled manner.

The composite random forest (CRF) is constructed using attribute pairs rather than individual attributes as in vanilla random forest (Breiman, 2001). Each attribute pair is composed of one attribute from each of the layers involved in dropout. CRF ranks all the attribute pairs based on their importance in determining the classification accuracy. Hence, the importance of an attribute pair captures how the interplay between the attributes affect classification accuracy. Consequently, the connections of the neural network that connects the attributes of an

**Corresponding author: Tel.: +91-33-2575 2913;
*Angshuman Paul is presently at National Institutes of Health, USA;
e-mail: bikash.santra@isical.ac.in (Bikash Santra)

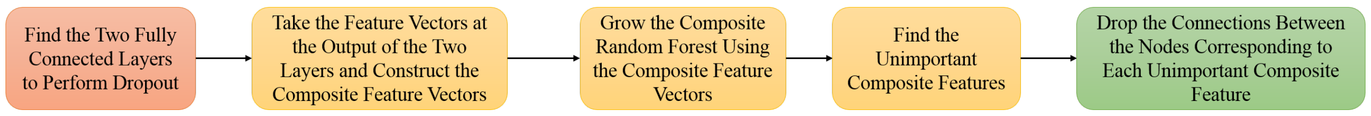
Find the Two Fully Connected Layers to Perform Dropout
Take the Feature Vectors at the Output of the Two Layers and Construct the Composite Feature Vectors
Grow the Composite Random Forest Using the Composite Feature Vectors
Find the Unimportant Composite Features
Drop the Connections Between the Nodes Corresponding to Each Unimportant Composite Feature

Figure 1: The overall flow of the proposed deterministic dropout.

important pair, are retained. These connections allow the propagation of class discriminative information. Other connections are dropped out. The block diagram of the proposed method is presented in Fig. 1.

Note that similar to the classical dropout (Hinton et al., 2012; Srivastava et al., 2014), our method also proposes dropout that prevents the networks from relying on particular combinations of attributes. In that sense our method preserves the basic tenets of dropout in preventing overfitting. Our method determines important connections analytically rather than existing approaches of dropping connections randomly. Since we remove unimportant connections through the proposed deterministic dropout, we can restrict the propagation of noise through the network and capture the salient information from training data. As a result, our method is well suited for noisy datasets. We validate this claim in Section 3.

The rest of the paper is organized as follows. In Section 2, we discuss the proposed method. The experiments and the results are presented in Section 3. We conclude the paper with the directions to future research in Section 4.

2. Methods

We first briefly discuss the random dropout procedure introduced in (Hinton et al., 2012) and (Srivastava et al., 2014). Consider a dataset $\mathtt{X}$ that is used as input to a neural network. Let $\mathtt{X}$ be composed of N number of data points $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$. Assume that there are M attributes associated with each data point. Hence, each input data point is an M dimensional feature vector. Now consider layer l of the neural network. Let $\mathtt{Y}^{[l]}$ be the input to the layer l and $\mathtt{Z}^{[l]}$ be the output from layer l. Assume that layer l and $(l+1)$ contain U and V numbers of neurons respectively. Then $\mathtt{Z}^{[l]}$ is the representation of N number of data points at the output of layer l. Clearly, $\mathtt{Z}^{[l]}$ contains data points each with U attributes. Hence each data point in $\mathtt{Z}^{[l]}$ is a U dimensional feature vector.

In a neural network, consider any two consecutive layers l and $(l+1)$. The output data from layer l is $\mathtt{Z}^{[l]}$ and let $\mathtt{Y}^{[l+1]}$ be the input data to layer $(l+1)$. Then

$$\mathtt{Y}^{[l+1]} = \mathtt{W}^{[l]}\mathtt{Z}^{[l]} + \mathbf{b}^{[l]}, \tag{1}$$

where $\mathtt{W}^{[l]}$ is the weight matrix connecting layers l and $(l+1)$ and $\mathbf{b}^{[l]}$ is the bias. Let $\varphi^{[l+1]}(\cdot)$ be the activation function associated with layer $(l+1)$. Then the output of layer $(l+1)$ is given by:

$$\mathtt{Z}^{[l+1]} = \varphi^{[l+1]}\left(\mathtt{Y}^{[l+1]}\right) = \varphi^{[l+1]}\left(\mathtt{W}^{[l]}\mathtt{Z}^{[l]} + \mathbf{b}^{[l]}\right). \tag{2}$$

In order to perform dropout, some connections between layers l and $(l+1)$ are to be terminated randomly. In (Srivastava et al., 2014), the terminations of connections are performed by creating a dropout mask $\mathtt{D}^{[l]}$, where the $(u, v)^{\text{th}}$ element $(u \in [1, U]\,;\ v \in [1, V])$ of $\mathtt{D}^{[l]}$ is $D^{[l]}(u, v) = \text{Bernouli}(p)$. Depending on the value of the threshold probability p, the value of $\mathtt{D}^{[l]}(u, v)$ would be either 1 or 0. Thus, dropout mask is created randomly. With dropout, (1) would be modified as:

$$\mathtt{Y}^{[l+1]} = \left(\mathtt{W}^{[l]} .\!* \mathtt{D}^{[l]}\right)\mathtt{Z}^{[l]} + \mathbf{b}^{[l]}, \tag{3}$$

where $.*$ is the Hadamard product which indicates term by term product of two matrices. Accordingly, from (2), the output of layer $(l+1)$ is:

$$\mathtt{Z}^{[l+1]} = \varphi^{[l+1]}\left(\left(\mathtt{W}^{[l]} .\!* \mathtt{D}^{[l]}\right)\mathtt{Z}^{[l]} + \mathbf{b}^{[l]}\right). \tag{4}$$

Next we discuss the motivation behind the proposed deterministic dropout for better generalization.

2.1. Motivation

Consider a network with mean squared classification error E. Also assume that w_{uv} is the weight connecting node u in layer l to node v in layer $(l+1)$. Then, according to (LeCun et al., 1990), the saliency of the connection is:

$$\Psi(u, v) = \frac{\partial^2 E}{\partial w_{uv}^2}\gamma^2(u, v), \tag{5}$$

where $\gamma(u, v)$ is a parameter that controls the connection. According to (LeCun et al., 1990), connections with high saliency improve the generalization of the network. Hence, we define the generalization ability of a network as:

$$G = \prod_{\forall (u,v)} \widehat{\Psi}(u, v), \tag{6}$$

where $\widehat{\Psi}(\cdot)$ is the normalized value of $\Psi(\cdot)$ in $[0, 1]$.

Assume that the importance measure for a connection depends on $\gamma(\cdot)$. For some connections, we have $\gamma(\cdot) \to 0$. Consequently, from (5), we have $\Psi(\cdot) \to 0$ for the connections with $\gamma(\cdot) \to 0$. If we retain the connections with $\gamma(\cdot) \to 0$, we get the generalization measure as G_1, $G_1 = \prod_{\forall (u,v)} \widehat{\Psi}(u, v)$.

Now suppose, we drop the connections, for which $\Psi(\cdot) \to 0$ (lower value of saliency). Then, the value of the generalization measure for the network after dropping the connections with low saliency is:

$$G_2 = \prod_{\forall (u,v) : \widehat{\Psi}(u,v) \nrightarrow 0} \widehat{\Psi}(u, v). \tag{7}$$

Clearly, $G_2 > G_1$. This indicates that dropping connections with low importance (and consequently low saliency) leads to better generalization. Hence, overfitting can be avoided by dropping connections with low importance. However, dropout method randomly drops connections without considering its

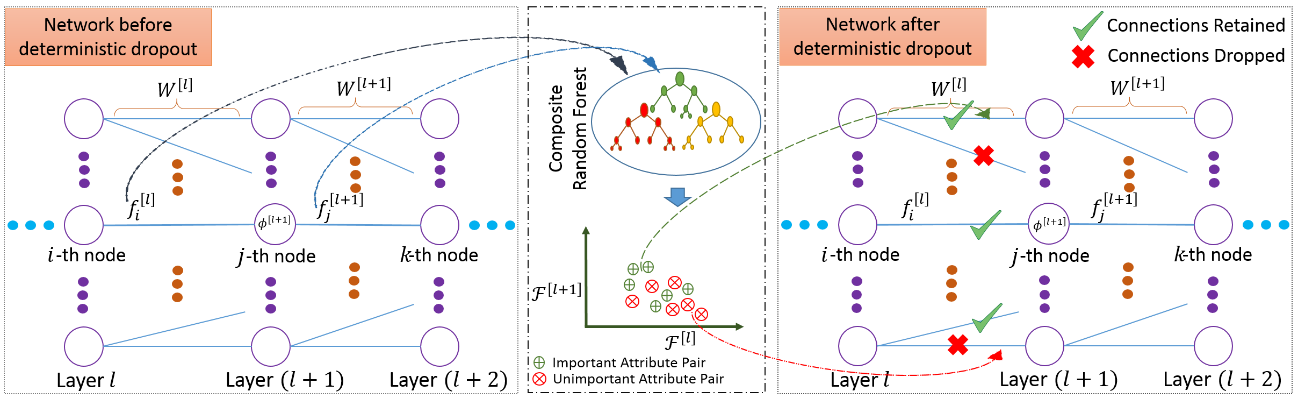

Figure 2: Deterministic dropout using composite random forest in a neural network. Blue straight lines (–): connections between nodes, $f_i^{[l]}$: the output from node i at the lth layer (referred to as attribute in the text), $\varphi^{[l+1]}$: activation function at the $(l+1)$th layer, $\mathtt{W}^{[l]}$: weight matrix at the lth layer.

importance. Next we present an analysis from the perspective of information-propagation on why we should drop unimportant connections.

Why Drop Unimportant Connections: Consider any two consecutive layers l and $(l+1)$ of a neural network. As discussed earlier, for each data point, $\mathtt{Z}^{[l]}$ (the output from layer l) is a U dimensional feature vector. Assume $\mathtt{Z}^{[l]}$ provides the feature vector $\mathcal{F}^{[l]}$ with U attributes $f_1^{[l]}, f_2^{[l]}, \ldots, f_U^{[l]}$ i.e. $\mathcal{F}^{[l]} = \left\{f_1^{[l]}, f_2^{[l]}, \ldots, f_U^{[l]}\right\}$. Similarly, assume that $\mathtt{Z}^{[l+1]}$ defines the feature vector $\mathcal{F}^{[l+1]}$ with V attributes $f_1^{[l+1]}, f_2^{[l+1]}, \ldots, f_V^{[l+1]}$ i.e. $\mathcal{F}^{[l+1]} = \left\{f_1^{[l+1]}, f_2^{[l+1]}, \ldots, f_V^{[l+1]}\right\}$. Since the connections between the layers are exhaustive, each attribute in layer $(l+1)$ is composed of the contributions from all the attributes in layer l.

However, in most of the practical scenarios, there will be several attributes in the output of layer l that do not contain class discriminative information (these attributes are termed as 'unimportant' attributes). As a result, there might be some attributes in the output of layer $(l+1)$ that also do not possess class discriminative information. This may in turn give rise to unimportant attributes in layer $(l+2)$ and so on. Thus, through these unimportant attributes, there might be a propagation of unimportant information till the last layer. This situation especially occurs in the cases of noisy datasets. Presence of unimportant attributes at the final layer may negatively affect the classification performance of the network. So, the propagation of unimportant information (through unimportant attributes) should be stopped as soon as it is detected.

We have already argued that each attribute in layer $(l+1)$ is composed of the contributions from all the attributes in layer l. So, in order to find and drop an unimportant connection, we need to look into the interplay between an attribute of layer l and an attribute of layer $(l+1)$. Hence, we have to find a method to look into the importance of attribute pairs composed of one attribute from layer l and one attribute from layer $(l+1)$. We design a composite random forest to take care of this problem. Once the unimportant connections are found, we drop those connections. A schematic of our method is presented in Fig. 2. Next we present the construction of the proposed composite random forest.

2.2. *Composite Random Forest*

Random forest (RF) can find the importance of different attributes in the original attribute space in a supervised manner (Paul et al., 2015; Breiman, 2001) exploiting the role of different attributes in classification. This is an advantage compared to other popular unsupervised techniques like principal component analysis (PCA) (Wold et al., 1987) and linear discriminant analysis (LDA) (Xanthopoulos et al., 2013).

In his original paper (Breiman, 2001), Breiman showed how to calculate the importance of different attributes individually using RF. It is also possible to find the importance of linear combinations of attributes. Nevertheless, the task of finding importance of pair of attributes is not straightforward. We need multivariate decision trees (Brodley and Utgoff, 1995) for this task. Typical multivariate decision trees take a subset of attributes and find an optimal separating hyperplane in the attribute subspace (Menze et al., 2011).

But our problem is different. We have a number of attribute pairs. In order to find importance of each such pair using RF, we need to construct an RF in which the nodes are split using a pair of the attributes. These require finding the winner attribute pair (from a randomly selected subset of attribute pairs) just as random forest finds an individual winner attribute (from a randomly selected subset of attributes). We also need to find a pair of values (one value corresponding to each attribute of the winner attribute pair) based on which the node will be split. For this, we design, what we call a 'composite random forest'. Before we discuss the construction of the composite random forest, let us first look at how nodes in the composite random forest will be split. This involves three steps: construction of composite attributes, finding the winner attribute pair and finding the split point. These steps are discussed next.

Construction of Composite Attributes: As we have seen, each data point at the output of layer l is a U dimensional feature vector $\mathcal{F}^{[l]}$. Similarly, we have a V dimensional feature vector $\mathcal{F}^{[l+1]}$ at the output of layer $(l+1)$. Note that $\mathtt{Z}^{[l]}$ (see (1)) contains N number of data points. Same is the number of data points for $\mathtt{Z}^{[l+1]}$.

Using the attributes from layers l and $(l+1)$, we first construct a composite set of attributes denoted as $\mathcal{F}^{[l,(l+1)]}$ by taking

Cartesian product of $\mathcal{F}^{[l]}$ and $\mathcal{F}^{[l+1]}$,

$$\begin{aligned}\mathcal{F}^{[l,(l+1)]} &= \mathcal{F}^{[l]} \times \mathcal{F}^{[l+1]} \\ &= \left\{(u, v) \mid u \in \mathcal{F}^{[l]} \text{ and } v \in \mathcal{F}^{[l+1]}\right\}. \end{aligned} \tag{8}$$

Next we grow a random forest using the composite attributes. Unlike the classical random forest (Breiman, 2001), each node in the proposed composite random forest is split using one of the composite attributes from $\mathcal{F}^{[l,(l+1)]}$. Recall that each element of the set $\mathcal{F}^{[l,(l+1)]}$ is actually an attribute pair. Hence, to grow the forest, we need to find the winner attribute pair from $\mathcal{F}^{[l,(l+1)]}$ for each node. Just like the random forest algorithm (Breiman, 2001), we first randomly select M' number of attribute pairs from $\mathcal{F}^{[l,(l+1)]}$. Out of those M' number of attribute pairs, we have to select the winner in each node. The process of finding the winner is discussed next.

Finding the Winner Attribute Pair: Each pair of attributes (u, v) obtained in (8) forms a 2D attribute subspace. If we plot all the data points in any such subspace, the data points form clusters. We use cluster properties to select the winner attribute pair in a node. Different methods (Zhao et al., 2017, 2018) have been proposed for defining the quality of clusters. We take the following strategy involving the clusters to find the winner attribute pair in a node.

Recall that in each node, we first randomly select M' attribute pairs. Hence, for each node, M' number of such subspaces are possible. We plot data points of that node in all of those subspaces. Consider any subspace constructed by an attribute pair. Assume the data points of class c form the c^{th} cluster in that subspace. Let the corresponding cluster center be $\alpha(c)$ and $r_i(c)$ be the distance of the i^{th} data point of cluster c from its center. If the cluster contains $N(c)$ number of data points, the average radius of cluster c is $r(c) = \frac{\sum_{\forall i \in c} r_i(c)}{N(c)}$. Similarly for the data points of class c', we have cluster center $\alpha(c')$ and radius $r(c')$. Then we define the separability between the above two clusters as:

$$\mathcal{S}(c, c') = \frac{||\alpha(c) - \alpha(c')||}{r(c) + r(c')}, \tag{9}$$

where $||\cdot||$ indicates the Euclidean distance. The higher the value of $\mathcal{S}(\cdot)$, the better the separation between the two clusters. Then the cluster separability index in the subspace (formed by the attribute pair (u, v), $u \in \mathcal{F}^{[l]}$ and $v \in \mathcal{F}^{[l+1]}$) is defined as the sum of separabilities between each pair of clusters:

$$\overline{\mathcal{S}(u, v)} = \sum_{\forall (c,c')} \mathcal{S}(c, c'). \tag{10}$$

It is quite obvious that the higher value of cluster separability index indicates better separability between the clusters in a subspace. This, in turn, means that data points of different classes are better-separated in a subspace with the higher value of cluster separability index. Hence, the best subspace is the one with the highest value of cluster separability index. The attribute pair that construct the best subspace is considered to be the winner attribute pair to split the node. Once the winner is found, the next task is to determine the threshold values of the pair of attributes on which the split will be performed. The pair of threshold values is termed as 'split point'.

Finding the Split Point: We assign weights to each cluster depending on its radius and the number of data points in it. Let $N(c)$ be the number of data points in cluster c. We define the weight of the cluster corresponding to the data points of class c as: $m(c) = \frac{N(c)}{r(c)}$.

In order to find the split point, we calculate the cluster weights in the subspace constructed by the winner attribute pair. The weighted centroid of the subspace is obtained using the cluster weights as:

$$\overline{\alpha} = \left(\sum_{\forall c} m(c)\alpha(c)\right) \Big/ \left(\sum_{\forall c} m(c)\right). \tag{11}$$

The weighted centroid is situated closer to the denser clusters (clusters with higher weights) and away from the scattered clusters (clusters with lower weights). Hence, the weighted centroid provides good partition of data points. So, we take the weighted centroid as the split point to split the node under consideration. Next we present the complete algorithm for the composite random forest.

Composite Random Forest Algorithm: In order to construct the composite random forest, we first find the composite feature vectors (following (8)) for all the training data points. Next, following (Breiman, 2001), we randomly choose N training data points with replacement from the training dataset to construct the bootstrap sample for growing a particular tree. Then at each node of the tree, we find the winner attribute pair based on the cluster separability index as presented in (10). Once the winner attribute is found, the split point is determined using (11). We split the node to create left child and right child nodes. The node splitting is continued until the number of data points in a node goes below a preset threshold. Repeating the above process for B trees in the composite random forest. We use this forest to find the importance of different attribute pairs.

Algorithm 1 Composite Random Forest

```
    Input: Number of trees $B$
    Output: Global importance of attribute pairs $\gamma_g(u, v)$
 1: procedure INITIALIZE ($\mathcal{F}^{[l,(l+1)]}$)
 2:     Given two sets of attributes $\mathcal{F}^{[l]}$ and $\mathcal{F}^{[l+1]}$, find the set
        of composite attributes $\mathcal{F}^{[l,(l+1)]}$ using (8).
 3:     Take $k = 1$.
 4: end procedure
 5: while $k \leq B$ do
 6:     while Nodes are left to be split do
 7:         Go to the next node to be split.
 8:         Select a random subset of attribute pairs from
            $F^{[l,(l+1)]}$.
 9:         From the random subset of attribute pairs, find the
            winner attribute pair by maximizing (10).
10:         Determine the split point in the winner attribute pair
            using (11).
11:         Based on the split point, create the left and the right
            child nodes.
12:     end while
13: end while
```

2.3. Finding the Importance of Attribute Pairs

The importance of different attribute pairs depends on two components. The first one is the local importance of an attribute pair inside a tree. The second one is the importance of the tree with respect to the entire forest.

Local Importance of an Attribute Pair: Consider the k^{th} tree Θ_k of the composite random forest. Let an attribute pair $(f_u^{[l]}, f_v^{[l+1]})$ be selected in the random subset of attributes for L_s number of nodes. Out of that, L_w times the attribute pair becomes the winner. Assume that in a node i of the k^{th} tree, the attribute pair $(f_u^{[l]}, f_v^{[l+1]})$ has been selected in the random subset of attributes. Also, let $\overline{S_i(u,v)}$ be the cluster separability index in the attribute subspace constructed by the attribute pair $(f_u^{[l]}, f_v^{[l+1]})$. Notably a cluster contains data of same classes. Further, assume that $(f_{u^*}^{[l]}, f_{v^*}^{[l+1]})$ is the winner attribute pair and $\overline{S_i(u^*,v^*)}$ is the cluster separability index in the attribute subspace constructed by the winner attribute pair. Then we define the local importance of the attribute pair $(f_u^{[l]}, f_v^{[l+1]})$ in tree Θ_k:

$$\gamma_k(u,v) = \left(\sum_{i=1}^{L_s} \frac{\overline{S_i(u,v)}}{\overline{S_i(u^*,v^*)}}\right) / L_s. \tag{12}$$

The above equation indicates, given the random selection, how successfully the attribute pair $(f_u^{[l]}, f_v^{[l+1]})$ became the winner attribute pair in different nodes. If the value of $\gamma_k(u,v) = 1$, it indicates that every time the attribute pair $(f_u^{[l]}, f_v^{[l+1]})$ was selected randomly in a node, the pair became the winner attribute pair. On the contrary $\gamma_k(u,v) = 0$ indicates that the attribute pair could not become the winner in any node. So, the higher the value of $\gamma_k(u,v)$, the more the importance of the attribute pair in tree Θ_k. If an attribute pair is not at all selected randomly for any node, we assign its local importance to be 1 to give the attribute pair a fair chance of being selected next.

Suppose some attribute pair has high local importance in a tree. But the classification performance of that tree is poor. So, in the context of the entire forest, that particular tree would be considered poor. Consequently, the attribute pairs that had high importance in that tree are actually not important with respect to the overall classification performance by the forest. So, while calculating the importance of the attribute pairs, we need to find the weights of individual trees as well. Next we discuss how to find the weights of individual trees and consequently find the global importance of an attribute pair.

Global Importance of an Attribute Pair: In order to calculate the global importance, we first need to find the weights of the individual trees. We calculate the weights of the individual trees $\beta(k)$ following (Paul et al., 2015). The global importance of the attribute pair $(f_u^{[l]}, f_v^{[l+1]})$ is:

$$\gamma_g(u,v) = \sum_{k=1}^{B} \gamma_k(u,v)\beta(k). \tag{13}$$

It is clear that an attribute pair with high value of $\gamma_g(\cdot)$ takes significant role in accurate classification. Hence, based on the global importance of attribute pairs, next we design the dropout mask for the neural network. Since the dropout mask is designed deterministically, this dropout mechanism is termed as deterministic dropout. The procedure of constructing composite random forest is presented in Algorithm 1.

2.4. Deterministic Dropout

Consider any two consecutive layers l and $(l+1)$, in between which we have to perform the dropout. We construct a composite random forest by taking the outputs from layers l and $(l+1)$. Let μ be the mean of global importance of the attribute pairs and σ be the standard deviation of the global importance.

Suppose we find that an attribute pair has low global importance. Then we drop the connection that connects the two attributes of that attribute pair. Let there be a connection from $f_u^{[l]}$ in layer l to $f_v^{[l+1]}$ in layer $(l+1)$. Assume that the corresponding mask value is $D^{[l]}(u,v)$. If we find that $\gamma_g(u,v)$ (the global importance of attribute pair $(f_u^{(l)}, f_v^{[l+1]})$) is low, we drop the connection from $f_u^{(l)}$ in layer l to $f_v^{[l+1]}$ in layer $(l+1)$. We implement this dropout by making $D^{[l]}(u,v) = 0$. If we do not need to drop the connection, we make $D^{[l]}(u,v) = 1$. In particular, if $\gamma_g(u,v) < (\mu - \kappa\sigma)$ (where κ is a constant), we say that the corresponding attribute pair has low importance. Subsequently, we drop the corresponding connection. Hence

$$D^{[l]}(u,v) = \begin{cases} 0 \text{ if } \gamma_g(u,v) < (\mu - \kappa\sigma), \\ 1 \text{ otherwise.} \end{cases} \tag{14}$$

Once the dropout mask is computed, we perform dropout using (3). Thus in our approach, dropping of unimportant connections result in better generalization of the network following (7). Note that at the time of testing, we use all the weights with their final values obtained during training. Next we discuss the performance of the proposed method.

3. Experiments & Results

Datasets & Parameters: We evaluate the performance of the proposed method from various different aspects. First, we examine the efficacy of the proposed method in noisy datasets with smaller number of training examples (refer Section 3.1). We use several noisy variants of MNIST (Variants, 2016) for this purpose. Second, we look into the effect of size of training data on our algorithm (refer Section 3.2). Next we evaluate the usefulness of the proposed method in benchmark datasets CIFAR-10 (Krizhevsky and Hinton, 2009), SVHN (Netzer et al., 2011) and Fashion MNIST (Xiao et al., 2017).

Finally, we use GroZi-120 (Merler et al., 2007) and Grocery Products (George and Floerkemeier, 2014) datasets for our experiments (refer Section 3.4). These datasets contain images of retail products. The training data in the above datasets consist of the images of individual products captured under *controlled studio environment* (Merler et al., 2007). On the contrary, the test data contains images of products which are cropped from images of racks captured in *uncontrolled supermarket environment* (Merler et al., 2007). Thus the training and the test data are acquired in different imaging environments. Furthermore, only a small number (typically 1 to 14) of training data are available per class for the above datasets. Hence the generalization capability of the proposed approach can be evaluated through the results on these retail product datasets. For the random forest, we take $B = 100$ trees. The value of κ in (14) is set to 0.5 The experiment for choice of κ is detailed in Appendix A. We

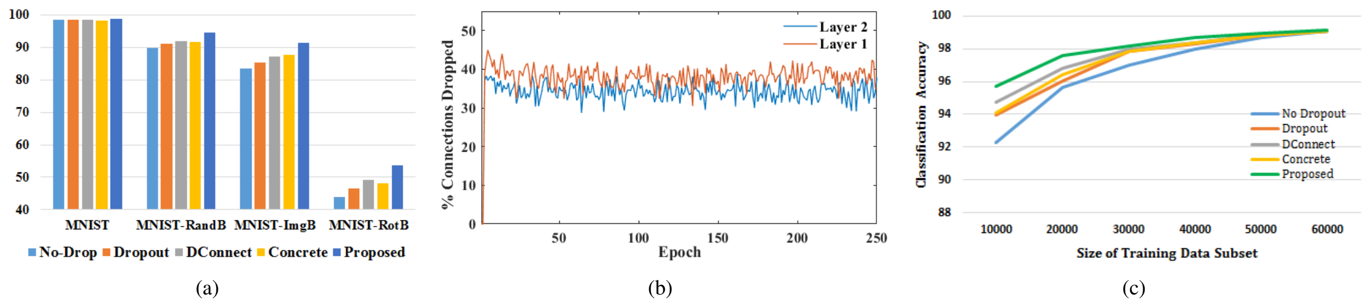

Figure 3: (a): Classification accuracies for MNIST and variants of MNIST using DNN$_1$. (b): Percentage of connections (w.r.t. total number of connections) dropped at each epoch for MNIST dataset using the proposed method. (c): Classification accuracies in MNIST dataset with increasing size of training subsets.

stop the splitting of a node when the number of data points in the node is less than 5. Experiments show that making nodes with smaller number of data points only increases computational burden without significantly improving the classification performance. Following (Breiman, 2001), we take $M' = \sqrt{M}$, where M is the total number of composite attributes.

Experimental Protocols: For our experiments, we train our networks for 250 epochs using mini-batch stochastic gradient descent with mini-batch size of 32 and momentum of 0.9. The learning rate is initialized at 0.01 and decreased by 10 times at every 25 epochs. For each dataset, we report the mean of the classification accuracies obtained by the five independent networks (Wan et al., 2013) which are trained using random permutations of the training data.

Competing Methods: We compare our method with a number of well-known regularization techniques such as dropout (Hinton et al., 2012), DropConnect (Wan et al., 2013) (abbreviated as DConnect) and concrete dropout (Gal et al., 2017) (abbreviated as Concrete). For each of the competing dropout methods, the dropout probability is set to 0.5 following (Krizhevsky et al., 2012). We also evaluate the performance of the networks without dropout (abbreviated as No-Drop). For all the competing approaches, we follow the same experimental protocols. Next we discuss the comparative results.

3.1. *Results on Noisy Data: Variants of MNIST*

We take several variants of MNIST (Variants, 2016). The variants are: MNIST-RandB (MNIST corrupted by random background), MNIST-ImgB (MNIST corrupted by background image) and MNIST-RotB (rotated MNIST with background images). The original MNIST (LeCun et al., 1998) dataset contains 60000 training and 10000 test examples. Whereas all of the above variants of MNIST are noisy and small (contain 12000 train and 50000 test images). The details of the noisy datasets is given in (Variants, 2016). For our experiments on MNIST (LeCun et al., 1998) and variants of MNIST (Variants, 2016), we take two different networks. The first one is a fully connected deep neural net (abbreviated as DNN$_1$). The second network is a convolutional neural network (abbreviated as CNN$_1$). In the following paragraphs, we describe the networks and analyze the results obtained using these networks.

Architecture of DNN$_1$: For this network, the input training and test images are normalized to fit into 20×20 pixels. The first layer is composed of 800 nodes (followed by an activation) and takes the image pixels as input. This layer is followed by a second fully connected layer of 800 nodes. The output of the second layer (after activation) is fed to a ten-class softmax classification layer. We implement the proposed deterministic dropout mechanism (and also the competing mechanisms) in between the first and second fully connected layers. We also use these dropout techniques between the second layer and the final softmax layer.

Analysis of Results Using DNN$_1$: The classification accuracies using different dropout mechanisms and without dropout have been presented in Fig. 3(a). Notice that the proposed deterministic dropout outperforms its competitors in all of the above datasets. However, the margin of improvement of the proposed method is more significant in the noisy datasets MNIST-RandB, MNIST-ImgB and MNIST-RotB by 2.82%, 4.19% and 8.94% respectively. Recall that each of the MNIST-RandB, MNIST-ImgB and MNIST-RotB datasets have only 12000 training example (in contrast to 60000 training examples in original MNIST). Also note that, all of the above datasets have signal to noise ratio (SNR) < -5 dB with respect to the original MNIST dataset (negative SNR indicates more noise than signal). Hence we conclude that our method outperforms its competitors by significant margin in case of noisy and small datasets.

Next we look into the number of connections dropped across different epochs. We plot the average numbers of dropped connections over different epochs at fully connected layer 1 and layer 2 in Fig. 3(b). Notice that the number of connections dropped at each iteration is not the same across different epochs. Since the total number of dropped connections vary across the epochs, we can say that same connections are not dropped at each epoch. Consequently, a different network is created at every epoch. Thus in our method, we are able to create an ensemble of networks which is the primary goal of dropout (Hinton et al., 2012; Srivastava et al., 2014). Further, since identical connections are not dropped in each epoch, we can conclude that the improvement in performance does not arise from the pruning of specific connections. Recall that the connections to be dropped in our method are determined by the importance of the connections as described in Section 2.4. Therefore, we can conclude that the performance improvement in our method is due to the quality of connections and not due to the number of the dropped connections. Next we look into the performance of the proposed deterministic dropout in a con-

Table 1: Mean classification accuracies (%) in variants of MNIST (using CNN_1), benchmark (using CNN_2) and retail product (using CNN_3) datasets

Methods	Variants of MNIST Datasets				Benchmark Datasets			Retail Product Datasets	
	MNIST	MNIST -RandB	MNIST -ImgB	MNIST -RotB	CIFAR-10	SVHN	Fashion MNIST	GroZi	Grocery
No-Drop	99.08	90.28	85.12	45.23	74.69	89.11	88.72	41.92	77.22
Dropout (Hinton et al., 2012)	99.04	91.65	86.18	48.69	75.36	90.30	90.03	41.30	78.37
DConnect (Wan et al., 2013)	99.10	92.82	88.20	49.92	74.92	90.55	89.66	41.66	78.30
Concrete (Gal et al., 2017)	99.06	92.53	89.63	49.88	75.27	90.40	90.10	42.36	78.02
Proposed	**99.14**	**95.22**	**92.48**	**54.48**	**76.35**	**91.61**	**91.13**	**45.15**	**81.62**

volutional neural network (CNN_1) for MNIST dataset.

Architecture of CNN_1: For this network, we take 24×24 input images during training and testing phases. Our network consists of a convolution layer of 32 activation maps. Each map is obtained using a 3×3 non-overlapping convolution mask. The convolution layer is followed by a *relu* activation layer and then a max pooling layer with 2×2 non-overlapping kernels. Then we have a second convolution layer of 64 activation maps using 5×5 overlapping convolution kernels. We place a *relu* activation layer and 2×2 overlapping max pooling layer after the second convolution layer. This is followed by a fully connected layer of 150 nodes where we use the proposed and competing dropout mechanisms. Finally, there is a log softmax output layer containing ten nodes each corresponding to a class label.

Analysis of Results Using CNN_1: The classification accuracies using CNN_1 are presented in Table 1. Notice that the proposed deterministic dropout achieves better classification accuracies compared to the other competing methods. As expected, the margin of improvement for our method in the noisy variant of MNIST is more significant compared to the improvement in MNIST. Thus the usefulness of the proposed deterministic dropout for noisy and small datasets is further justified even for CNN.

3.2. *Effect of the Size of Training Dataset*

We take the original MNIST dataset that contains 60000 training examples. We train CNN_1 with a smaller subset of 10000 examples of training data and evaluate the performance of the network using classical dropout and the proposed method. We repeat this experiment with gradually increasing size of training subsets. In Fig. 3(c), we plot the classification accuracies of dropout and proposed method as the training data subset size increases. Notice that the relative improvement of performance of the proposed method compared to the classical dropout is more significant for smaller training datasets.

3.3. *Results on Benchmark Datasets*

Next we evaluate the efficacy of the proposed method in benchmark datasets CIFAR-10, SVHN and Fashion MNIST. For this purpose, we use a convolutional neural network (CNN_2) whose architecture is presented next.

Architecture of CNN_2: Since CIFAR-10 & SVHN datasets contain $32 \times 32 \times 3$ RGB images (28×28 grayscale images for Fashion MNIST), the first convolution layer of CNN_2 is designed using a $3 \times 3 \times 3$ convolution mask. There are three convolution layers (each followed by a relu and a max pooling layer) in this network with 32, 64 and 128 activation maps respectively. The third convolution layer is followed by two fully connected layers of 80 and 10 nodes respectively in between which we apply different dropout mechanisms.

Results & Analysis: For our experiments, we follow the same protocol as the one we have followed for experiments on MNIST. The comparative performances of different dropout methods for benchmark datasets are presented in Table 1. Notice that compared to the other approaches, the proposed method achieves better classification accuracies in CIFAR-10, SVHN and fashion MNIST datasets.

3.4. *Results on Retail Product Datasets*

For the two retail product datasets (GroZi-120 (Merler et al., 2007) and Grocery Products (Grocery) (George and Floerkemeier, 2014)), we show the performance of the proposed method using a convolutional neural network (CNN_3). We first present the architecture of the network followed by experimental results.

Architecture of CNN_3: The architecture of CNN_3 is similar to that of AlexNet (Krizhevsky et al., 2012) except the output layer. The output layer of CNN_3 contains as many nodes as the number of classes in the dataset under consideration. The weights of CNN_3 network (except the output layer) are initialized with that of pre-trained AlexNet. Like AlexNet, in the architecture of CNN_3, classical dropout (Hinton et al., 2012) is applied in the first two fully connected layers. We apply the proposed deterministic dropout and the other competing methods in CNN_3 as a replacement of the classical dropout.

Results & Analysis: All the images of GroZi-120 and Grocery Products datasets are resized to $224\times224\times3$. In these datasets, the number of training images per class is in between 1 and 14. Hence, we perform data augmentation following (Perez and Wang, 2017). For each class, almost $\sim 10^4$ train images are augmented in order to train CNN_3. The classification accuracies for different retail product datasets are presented in Table 1. It can be seen that the proposed method outperforms all its competitors in both the datasets. In particular, the percentage of improvement in classification accuracy by the proposed method (compared to its closest competitor) is 6.5% for the GroZi-120 dataset and 4.1% for the Grocery Products dataset. The superiority in the performance of the proposed method is due to its capability of better generalization. As a result, although the training data and the test data are captured in different environments, our method outperforms its competitors.

3.5. *Notes on Training Time*

The training of the proposed deterministic dropout involves construction of a random forest. This causes increase in training time. As mentioned in Section 3, we take 100 trees for growing a random forest in our implementation. The average depth of the trees in the forest is 39. The construction of a decision tree of depth 39 takes only 1.09s. Our model is implemented in python and tested in a computing system with 64GB RAM, Intel Core i7-7700K CPU @ 4.2GHz×8 and GeForce GTX TITAN 6GB GPU. Thus, in a fully parallel computing architecture, the construction of random forest also takes about 1s in each epoch. This marginal increase in training time comes with the benefit of significantly improved classification performances, as evident from Fig. 3 and Table 1. Further, the test time in the proposed method is not affected since we do not construct random forest during test. Therefore we can say that the proposed method does not add computational delays at test time.

4. Conclusions & Future Work

We propose a method that deterministically identify and terminate the unimportant connections in a neural network. A composite random forest is used for finding the unimportant connections. Results on a number of different datasets establish the superiority of the proposed method. The performance of our method is found to be superior in noisy and smaller datasets. In the future, we would aim to generalize the deterministic dropout strategy over the entire network including the convolution layers instead of specific fully connected layers.

References

Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

Brodley, C.E., Utgoff, P.E., 1995. Multivariate decision trees. Machine learning 19, 45–77.

Calvo-Zaragoza, J., Toselli, A.H., Vidal, E., 2019. Handwritten music recognition for mensural notation with convolutional recurrent neural networks. Pattern Recognition Letters URL: `http://www.sciencedirect.com/science/article/pii/S0167865519302338`, doi:`https://doi.org/10.1016/j.patrec.2019.08.021`.

Cheng, E.J., Chou, K.P., Rajora, S., Jin, B.H., Tanveer, M., Lin, C.T., Young, K.Y., Lin, W.C., Prasad, M., 2019. Deep sparse representation classifier for facial recognition and detection system. Pattern Recognition Letters 125, 71–77.

Gal, Y., Hron, J., Kendall, A., 2017. Concrete dropout, in: Advances in Neural Information Processing Systems, pp. 3581–3590.

George, M., Floerkemeier, C., 2014. Recognizing products: A per-exemplar multi-label image classification approach, in: European Conference on Computer Vision, Springer. pp. 440–455.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 .

Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images .

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, pp. 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

LeCun, Y., Denker, J.S., Solla, S.A., 1990. Optimal brain damage, in: Advances in neural information processing systems, pp. 598–605.

Menze, B.H., Kelm, B.M., Splitthoff, D.N., Koethe, U., Hamprecht, F.A., 2011. On oblique random forests, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer. pp. 453–469.

Merler, M., Galleguillos, C., Belongie, S., 2007. Recognizing groceries in situ using in vitro training data, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE. pp. 1–8.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y., 2011. Reading digits in natural images with unsupervised feature learning, in: NIPS workshop on deep learning and unsupervised feature learning, p. 5.

Paul, A., Dey, A., Mukherjee, D.P., Sivaswamy, J., Tourani, V., 2015. Regenerative random forest with automatic feature selection to detect mitosis in histopathological breast cancer images, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015. Springer, pp. 94–102.

Perez, L., Wang, J., 2017. The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621 .

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15, 1929–1958.

Variants, M., 2016. Mnist noisy variants. `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DeepVsShallowComparisonICML2007`.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R., 2013. Regularization of neural networks using dropconnect, in: International Conference on Machine Learning, pp. 1058–1066.

Wold, S., Esbensen, K., Geladi, P., 1987. Principal component analysis. Chemometrics and intelligent laboratory systems 2, 37–52.

Xanthopoulos, P., Pardalos, P.M., Trafalis, T.B., 2013. Linear discriminant analysis, in: Robust data mining. Springer, pp. 27–33.

Xiao, H., Rasul, K., Vollgraf, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 .

Zhao, C., Chen, K., Wei, Z., Chen, Y., Miao, D., Wang, W., 2019. Multilevel triplet deep learning model for person re-identification. Pattern Recognition Letters 117, 161 – 168. URL: `http://www.sciencedirect.com/science/article/pii/S0167865518301491`, doi:`https://doi.org/10.1016/j.patrec.2018.04.029`.

Zhao, C., Wang, X., Miao, D., Wang, H., Zheng, W., Xu, Y., Zhang, D., 2018. Maximal granularity structure and generalized multi-view discriminant analysis for person re-identification. Pattern Recognition 79, 79–96.

Zhao, C., Wang, X., Wong, W.K., Zheng, W., Yang, J., Miao, D., 2017. Multiple metric learning based on bar-shape descriptor for person re-identification. Pattern Recognition 71, 218–234.

Appendix A. Choice of κ

In order to choose the value of κ, we evaluate the classification performances by varying κ. The classification performances of different networks, experimented on various datasets, by varying κ in the proposed method are presented in Table A.2. Notice that in most cases, we get the best classification accuracy with $\kappa = 0.5$. Therefore, we choose $\kappa = 0.5$ for our experiments.

Table A.2: Classification accuracies of different networks on various datasets by varying κ

Datasets	κ								
	0.00	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00
DNN_1 on MNIST	98.52	98.78	**98.80**	98.73	98.72	98.67	98.65	98.65	98.63
DNN_1 on MNIST-RandB	94.30	94.60	94.60	**94.62**	94.57	94.50	94.52	94.55	94.55
DNN_1 on MNIST-ImgB	91.39	91.46	**91.48**	91.46	91.45	91.45	91.30	91.35	91.28
DNN_1 on MNIST-Rot	90.10	90.29	**90.30**	90.21	90.24	90.26	90.26	90.25	90.26
DNN_1 on MNIST-RotB	53.40	53.20	**53.60**	53.30	53.25	53.35	53.20	53.40	53.25
CNN_1 on MNIST	99.10	99.12	**99.14**	99.00	99.09	99.12	99.10	99.11	99.13
CNN_1 on MNIST-RandB	95.13	95.15	**95.22**	95.21	95.20	95.10	95.19	95.20	95.17
CNN_1 on MNIST-ImgB	**92.49**	92.47	92.48	92.48	**92.49**	92.40	92.38	92.36	92.30
CNN_1 on MNIST-RotB	54.19	54.34	**54.48**	54.36	54.40	54.30	54.32	54.35	54.35
CNN_2 on CIFAR 10	76.25	**76.35**	**76.35**	**76.35**	76.29	76.25	76.28	76.25	76.00
CNN_2 on SVHN	91.41	91.53	**91.61**	91.60	91.60	91.50	91.41	91.14	90.89
CNN_2 on Fashion MNIST	90.90	90.99	**91.13**	90.40	90.49	90.10	90.30	90.60	90.10
CNN_3 on GroZi	44.15	44.20	**45.15**	45.00	44.45	44.00	44.20	44.15	43.45
CNN_3 on Grocery	80.00	80.10	**81.62**	81.40	81.00	80.94	80.90	80.60	79.95