Enhancing feature augmentation of Deep Forests

Marcel Aguilar Garcia, ID: 20235620
m.aguilargarcia1@nuigalway.ie

9th April 2021

1 Introduction

Deep Learning has gained a lot of popularity and the usage of Deep Neural Networks (DNNs) has seen an increase in many applications such as image and speech recognition. DNNs have been proven to outperform other machine learning models in these applications [1]. However, this superior performance usually comes with more complexity and this has brought some challenges. To begin with, Neural Network architectures have to be defined in advanced, which means that its complexity is determined before the training phase. In general, DNNs have a large amount of hyper-parameters and they are known to be very sensitive to them. Additionally, they are difficult to be interpreted and are usually seen as black-box models. Finally, small data sets do not work well with DNNs and a large amount of data is required to achieve good performance [2].

While Deep Learning has always been seen as part of Neural Networks (NN), a recent study from Zhi-Hua Zhou and Ji Feng [2] has explored the idea of building deep models that are not based on NN. They believe that most of the success of DNNs comes from the layer-by-layer processing and that shallow models can be used to achieve deep models. In this way, they have introduced the idea of Deep Forest (DF) which is based on a cascade structure of shallow models, specifically, decision tree forests. Additionally, Deep Forests have been enhanced in order to handle image processing by using a procedure of multi-grained scanning.

The results from this study show that DFs are already able to provide similar results than well-known DNNs [2]. On top of that, DFs have many advantages over traditional DNNs. To start with, DFs have less hyper-parameters to which they are more robust. Moreover, as the algorithm is able to stop when the accuracy achieves a threshold, the cascade architecture does not have to be determined in advanced. In consequence, DFs have lower complexity than DNNs and are more suitable for tasks with small data sets. Because of these advantages, DFs have already been used in many real life tasks [3][4][5][6].

However, it is undeniable than DNNs are more mature than DFs and that additional experiments and improvements have be done to the original algorithm. Furthermore, DNNs are able to perform better than DFs in some image recognition tasks [2]. In the following section I describe and categorise some of the work that has followed up with DFs.

2 Related Work

The original algorithm of Deep Forests was published in 2019, which makes this algorithm relatively new compared to traditional DNNs. In consequence, there are not many papers related to DFs yet. In general, the improvements that have been done to DFs can be split into three categories. One category that is based on improving the issues related to large scale data and time performance. A second category that enhances the algorithm for a specific type of data (e.g. for imbalance data). Finally, a third category, which is trying to improve the prediction performance of DFs by changing the original algorithm.

Example of studies from the first category are [7] and [8]. The authors of [7] claim that Deep Forests are inefficient and lack scalability. A new version of the algorithm, ForestLayer, is presented which is built

on distributed task-parallel platforms. ForestLayer reduces the training time of DFs and increases accuracy in the case of large data sets. In contrast, the authors of [8] suggest a modification of the original algorithm that uses Hashing Learning and transforms data into low dimensional representations.

Examples of the second category are [9] and [10]. The authors of [9] have proposed a new version of DFs, Imbalanced Deep Forest, that is able to achieve better results in small datasets with imbalanced data. In a similar way, the authors of [10] are presenting a new version of DFs, Multi-label deep forest, that has been adapted for Multi-Label learning in which more than one label can be assigned to each instance.

Examples of the third category are [11], [12], and [13]. Motivated by AdaBoost algorithm, the authors of [13] have modified DFs to use weighted instances. These weights represent the probability of an instance being miss-classified. In a similar way, and motivated by the idea of metric learning, the authors of [11], suggest a modification that uses weights. However, in this case, the weights are used to measure the differences of the vectors returned by each forest. The more different the forests are, the better the accuracy of the DFs will be. Finally, in [12], the authors are following a similar approach than the one suggested in the original paper and are increasing the accuracy of DFs by modifying the class distribution vectors.

3 Research Questions

The original paper [2] suggests that enhancing data augmentation of Deep Forests by adding other leaf class distributions such as sibling and parent nodes, could lead to better results. Similar ideas have been done such as [12] which have been proven to improve the accuracy of the model. However the exact modification suggested in the original paper have not yet been studied. In order to prove that this modification improves accuracy, the algorithm should be adapted to the new vectors and experiments should be done to compare the performance with existing results. Having this in mind, it seems reasonable to ask the following questions:

1. Would Deep Forest accuracy increase by adding other class distributions?
2. What is impact on time and memory performance by doing such a change?
3. Does the risk of overfitting increases?
4. Does this modification work better in specific types of data?

The following section introduces Deep Forests and expands the idea of adding additional leaves class distributions.

4 Data Augmentation in Deep Forests

In the original Deep Forest, each level of the cascade is composed of two completely-random tree forests and two random forests. Given an example, each decision tree forests will return an n-dimensional vector with the estimate class distribution. Then, the input is passed to the next level together with these vectors. While this is considered a simple feature augmentation, [2] results have shown that is able to compete and return similar results than well-known DNNs.

Data augmentation is often used to increase accuracy in DNNs [14][1]. As the authors from [2] suggest, it seems reasonable to think that the same approach could be used to improve the accuracy of Deep Forests. In order to understand how data augmentation can be enhanced in this algorithm, there should be first a clear understanding on how trees and forests are calculating the class probabilities of an example:

4.1 Decision Trees

Let's consider a classification task with C classes that uses a decision tree $\mathcal{T}$ to classify an example $v = (f_1, \ldots, f_n)$, where n is the number of features and f_i the value of feature i for $i = 1, \ldots, n$. Through a sequence of binary decisions, the example is going to fall in one of the leaves from the tree. The decision tree is going to predict that example as the majority class in that leaf. Additionally, the tree can return the probabilities of each class by giving the frequencies of all classes from that leaf. This can be represented as a vector $P = (P_1, \ldots, P_C)$,

where P_i is the probability of an example being classified as class i.

Figure 1 shows the diagram of a decision tree for a binary classification task. This tree has four leaves and, given an example, a prediction will be done based on the leaf where the example falls in. For example, Leaf 1 has two samples of *yellow* class and one sample of *blue* class. This means that an example that is allocated to this leaf, will be predicted as *yellow* with a probability of 66.6%. This is equivalent to say that the class distribution vector for Leaf 1 is $P \approx (0.67, 0.33)$.

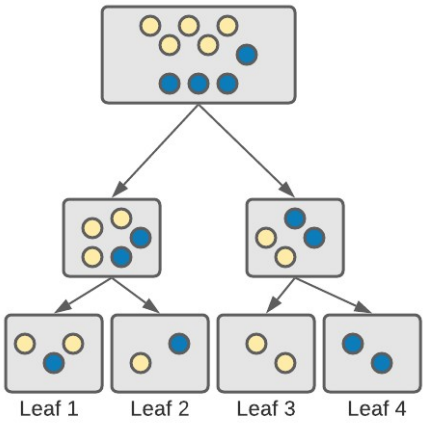

Figure 1: Decision Tree Classifier with two classes

4.2 Decision Tree Forests

A decision tree forest can be seen as a collection of decision trees $\mathcal{F} = \{\mathcal{T}_1, \ldots, \mathcal{T}_N\}$. A forest $\mathcal{F}$ will predict an example as the majority class from the predictions given by its trees. In the same way, a decision tree forest can return the probabilities for each class by averaging the probabilities of all trees. Let $P^{\mathcal{T}_i} = (P_1^{\mathcal{T}_i}, \ldots, P_C^{\mathcal{T}_i})$ be the vector representing the class distribution of a tree $\mathcal{T}_i$ then, the class distribution of the forest $\mathcal{F}$ can be calculated as:

$$P^{\mathcal{F}} = \left(\frac{\sum_{i=1}^{N} P_1^{\mathcal{T}_i}}{N}, \ldots, \frac{\sum_{i=1}^{N} P_C^{\mathcal{T}_i}}{N} \right) \tag{1}$$

Figure 2 shows a Forest with five trees, the prediction for each tree and its class distribution. The majority class predicted by the trees is *yellow*. Additionally, the class distribution of the forest can be calculated by averaging the distributions of the trees:

$$P^{\mathcal{F}} = \left(\frac{0.72 + 0.45 + 0.67 + 0.86 + 0.51}{5}, \frac{0.28 + 0.55 + 0.33 + 0.14 + 0.49}{5} \right) \approx (0.64.0.36) \tag{2}$$

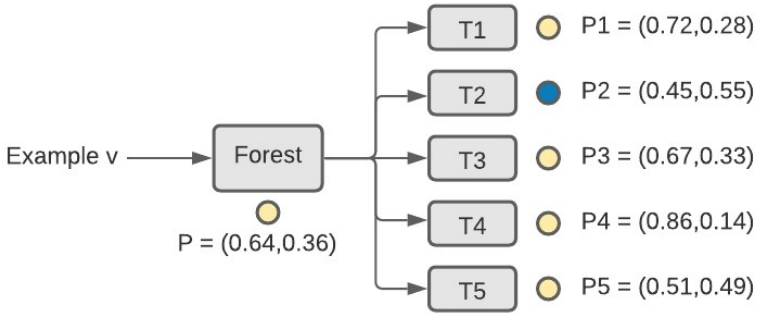

Figure 2: Decision Tree Forest with five trees

4.3 Deep Forests

In a Deep Forest, each cascade has four forests returning a vector $P^{\mathcal{F}}$. Each level returns the input concatenated with these vectors. Therefore, the output of a layer is given by the vector:

$$P^{\mathcal{C}} = (v, P^{\mathcal{F}_1}, P^{\mathcal{F}_2}, P^{\mathcal{F}_3} P^{\mathcal{F}_4}) \tag{3}$$

where v is the input vector and $P^{\mathcal{F}_i}$ is the class distribution of forest $\mathcal{F}^i$ for $1 \leq i \leq 4$.

Figure 3 shows the first level of a Deep Forest with four decision tree forests. The output from Level 1 is the concatenation of the input v and all forest class distributions. In this example, this output is $P^{\mathcal{C}_1} = (v, 0.9, 0.1, 0.51, 0.49, 0.22, 0.78, 0.63, 0.7)$.

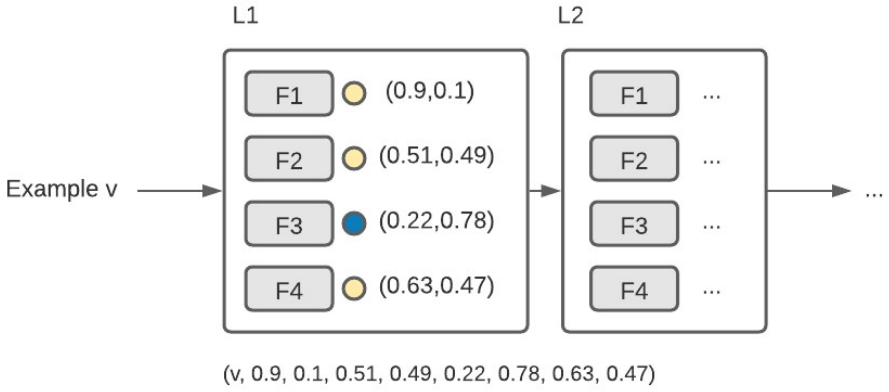

Figure 3: Level 1 from a Deep Forest with four forest

In the first level, $P^{\mathcal{C}_1}$ will be a vector of length $|v| + \sum_{i=1}^{4} |P^{\mathcal{F}_i}| = n + 4C$ where n is the number of features and C is the total number of classes. The length will increase to $n + 8C$ in the second level, $n + 12C$ in the third, etc. In general, L-th level should return a vector of length $|P^{\mathcal{C}_L}| = n + 4LC$.

The original Deep Forest from [2] is using this vector from Equation (3) to augment information. However, the authors believe that including the class distribution of parent nodes, sibling nodes, or other relevant features, may lead to better results. Recall that a tree can be seen as a sequence of binary decisions. The

sibling node of a leaf, is the node that comes from taking the complementary decision in the parent node. In Figure 1, Leaf 1 has a class distribution of $(0.67, 0.33)$, its sibling leaf, Leaf 2, has a class distribution of $(0.5, 0.5)$. In order to enhance the feature augmentation of a deep forest by adding the sibling nodes class distribution, the vector returned by this tree would be the concatenation of these two vectors. In this example, the vector returned should be $(0.67, 0.34, 0.5, 0.5)$.

In general, in order to enhance feature augmentation by adding the class distribution of an additional leaf, a decision tree should return the vector:

$$P^{\mathcal{T}} = (P_1, \ldots, P_n, P'_1, \ldots, P'_n) \tag{4}$$

where $P' = (P'_1, \ldots, P'_C)$ is class distribution of the leaf being considered.

Note that it would be possible to have multiple class distributions on the same vector, e.g., if considering sibling and parent nodes at the same time, the vector should include both distributions:

$$P^{\mathcal{T}} = (P_1, \ldots, P_n, P'_1, \ldots, P'_n, P''_1, \ldots, P''_n) \tag{5}$$

For simplicity, this research will just consider one class distribution.

In a similar way, taking Equation (4) into account, each forest should be modified to return:

$$P^{\mathcal{F}} = \left(\frac{\sum_{i=1}^{N} P_1^{\mathcal{T}_i}}{N}, \ldots, \frac{\sum_{i=1}^{N} P_C^{\mathcal{T}_i}}{N}, \frac{\sum_{i=1}^{N} P_1'^{\mathcal{T}_i}}{N}, \ldots, \frac{\sum_{i=1}^{N} P_C'^{\mathcal{T}_i}}{N} \right) \tag{6}$$

In this case, the length of $P^{\mathcal{C}_1}$ will be $|v| + \sum_{i=1}^{4} |P^{\mathcal{F}_i}| = n + 8C$. The length will increase to $n + 16C$ in the second level, $n + 24C$ in the third, etc. In general L-th level should return a vector of length $|P^{\mathcal{C}_L}| = n + 8LC$. Therefore, this new vector has $2C$ more features compared to the original one.

The information explained in this section shows that, in order to enhance data augmentation for Deep Forests, then:

1. Decision Tree algorithm should be modified in order to return the vector from Equation (4).
2. Decision Tree Forest algorithm should be modified in order to return the averages from Equation (6).
3. Deep Forests should be using these modified versions for trees and forests.

5 Research Method

The chosen subject is relatively new and there are not many papers related to Deep Forests yet. This subject is a combination of Deep Neural Networks and Ensemble Methods, in particular, Random Forests.

The search for DNNs was done to have a good understanding of the theory and latest improvements for DNNs. The search terms used for DNNs were "Deep Learning" and "Deep Neural Networks". In order to have the latest information only papers that were published after 2017 were considered. The search was done using NUI Galway Library and Google Scholar. Related Subjects were then used to find similar articles.

In a similar way, the search terms used for Ensemble Methods were "Ensemble Methods", "Ensemble Learning", and "Random Forest". There was no need to limit the year as Random Forest still uses an algorithm similar to original one from 2001. While the search could have been narrowed to just Random Forests, other ensemble methods could be used to build similar deep models (e.g. Gradient Boosting Classifier) which can motivate future ideas. The search was focused on surveys that were explaining the mathematics and theory behind random forests and ensemble methods in general.

Finally, the search term "Deep Forest" was used to find papers related to the subject itself. These papers were split into two categories, the ones that were improving the algorithm and use cases in which DFs have applied.

6 Conclusions

Deep Forests were built using the main characteristics from DNNs [2]. The fact that data augmentation can improve accuracy of DNNs [14], seems to be a good basis for thinking that the same could happen with Deep Forests. This statement can be supported by similar studies such as [13] that have proven to increase the accuracy by modifying the class distribution vectors.

In this research, I have explained some of the published papers that have presented modifications to Deep Forest and I have categorised the different modifications. I have shown that, while some of this publications have focused on ideas similar to the suggested in the original paper, the suggestion has not yet been implemented and tested. I have shown that [2] and [13] are supporting that this idea will probably show a better performance in DFs. Additionally, I have explained how decision trees and decision tree forests algorithms can be modified in order to improve data augmentation for Deep Forests. As suggested in Section 3, it would be interesting to find an alternative that does not use class distribution. This would avoid having a low number of augmented features for classification tasks with very few classes (e.g. binary).

It is important to note that any modification for data augmentation can be done in the way that this research has presented. By first expanding the class distribution vectors of each tree, then forests, and finally Deep Forests.

The same experiments that were used in [2] could be used to compare the enhanced version of Deep Forests with the original algorithm. As the enhanced version will be using longer vectors in each layer, it is expected that this could have an impact on time and memory performance. It is suggested that, if the algorithm is proven to increase accuracy, additional tests are done to cover this.