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 [2][3]. To begin with, Neural Network architectures have to be defined in advanced, which means that their complexity is determined before the training phase [3][4]. In general, DNNs have a large amount of hyper-parameters and are known to be very sensitive to them [5][6]. Additionally, they are difficult to be interpreted and are usually seen as black-box models [7][8]. Finally, small data sets do not work well with DNNs and a large amount of data is required to achieve good performance [9][10][11].

While Deep Learning has always been seen as part of Neural Networks (NN), a recent study from Zhi-Hua Zhou and Ji Feng has explored the idea of building deep models that are not based on NN [3]. 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, DFs have been enhanced in order to handle image processing by using a procedure of multi-grained scanning [3].

DFs have been shown to provide similar results to those obtained through well-known DNNs [3]. On top of that, DFs have many advantages over traditional DNNs [3]. To start with, DFs have less hyper-parameters to which they are more robust. Moreover, the algorithm is able to stop when an accuracy threshold is reached which means that 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 [12][13][14][15].

In fields such as healthcare and finance where interpretability and robustness of the models are important, high performance of DNNs may not be enough [16][17]. Improving accuracy of DFs could give an alternative of a model with high performance without having any of these disadvantages. Unfortunately, DNNs are still able to perform better than DFs in some tasks [3]. In general, it is undeniable that DNNs are more mature than DFs and that further research have to be done.

In the following section, I explain the research method used in this literature review.

2 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. A first one with the papers that are trying to improve the algorithm and a second category of real scenarios where DFs have been applied.

In the following section, I summarise and categorise some of the improvements and work that has followed up with DFs.

3 Related Work

The original algorithm of Deep Forests was published in 2018, 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 focused 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 imbalanced data). Finally, a third category, that focuses on improving the general prediction performance by modifying the original algorithm.

Within the first category it has been argued 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 [18]. On the other hand, a different approach has been suggested using Hashing Learning to transform data into low dimensional representations [19].

Within the second category, it has been proposed a new version of DFs, Imbalanced Deep Forest, that is able to achieve better results in small datasets with imbalanced data [20]. In a similar way, other studies have presented a new version of DFs, Multi-label deep forest, that has been adapted to Multi-Label learning in which more than one label can be assigned to each instance [21].

Within the third category, motivated by Adaboost algorithm, some scholars have proposed a modification of DFs that uses weighted instances [22]. These weights represent the probability of an instance being miss-classified. Another modification that uses weights has been suggested to measure the differences of the vectors returned by each forest to increase accuracy [23]. Finally, a similar approach than the one suggested in the original paper [3] was able to increase the accuracy of DFs by modifying the probabilities from the class distribution vectors [24].

The following section opens some research questions based on the findings of the third category.

4 Research Questions

The original paper 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 [3]. While similar ideas have been explored [24], the exact modification suggested by the original paper has not yet been done. 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 this modification increase the risk of overfitting?
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.

5 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 [3]. In order to understand how data augmentation can be enhanced, there should be first a clear understanding on how trees and forests are calculating the class probabilities of an example:

5.1 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 $P \approx (0.67, 0.33)$.

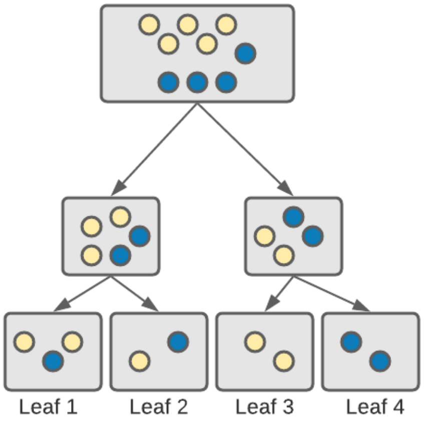

Figure 1: Decision Tree Classifier with two classes

5.2 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:

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

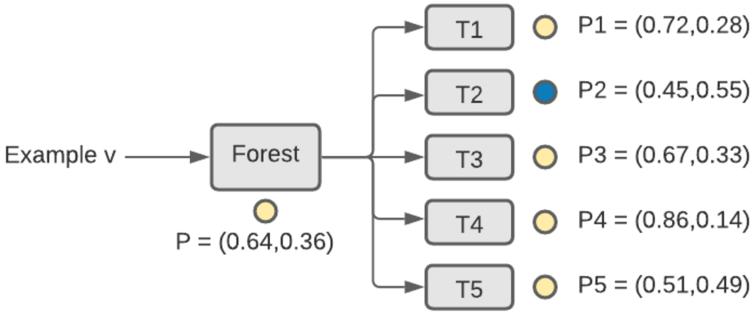

Figure 2: Decision Tree Forest with five trees

5.3 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 $P^{\mathcal{C}_1} = (v, 0.9, 0.1, 0.51, 0.49, 0.22, 0.78, 0.63, 0.7)$.

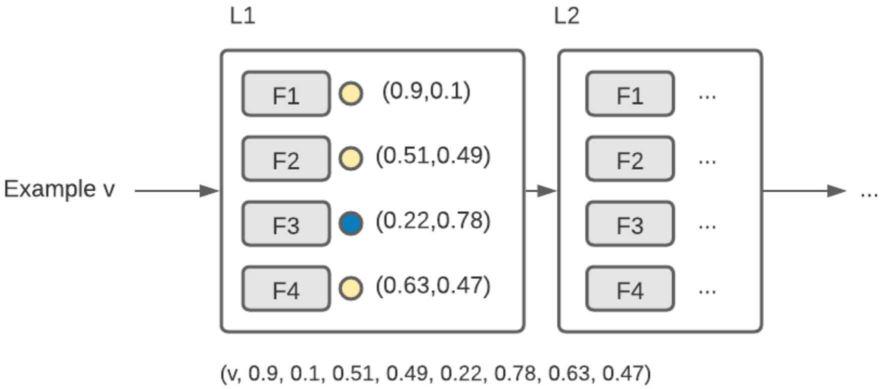

Figure 3: Level 1 from a Deep Forest with four forest

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 uses 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.

6 Conclusions

Deep Forests have been shown to have an impressive performance that is comparable to well-known DNNs. Since the original publication in 2019, many improvements and enhancements have been presented which have considerably reduce the time performance and increase accuracy of DFs. In this literature review, I have explained and categorised most of the modifications that have been done to the original algorithm. In particular, I have shown how some of these modifications are able to increase the general performance by changing the class distribution vectors of DFs.

The proponents of DFs suggested that using data augmentation with other class distributions could increase the accuracy of the model. Additionally, this has been supported by the results that similar modifications have achieved [24]. Section 3 introduces the main questions that could be used in order to follow up with this modification. Section 4 presents an introduction to DFs and adds the detail on how this enhancement could be used in the algorithm.

It is important to note that any modification for data augmentation based on the class distribution 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 [3] 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.

References

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[2] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[3] Z.-H. Zhou and J. Feng, "Deep forest," *National science review*, vol. 6, no. 1, pp. 74–86, 2019.

[4] A. Fiszelew, P. Britos, A. Ochoa, H. Merlino, E. Fernández, and R. García-Martínez, "Finding optimal neural network architecture using genetic algorithms," *Advances in computer science and engineering research in computing science*, vol. 27, pp. 15–24, 2007.

[5] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[6] M. Bashiri and A. F. Geranmayeh, "Tuning the parameters of an artificial neural network using central composite design and genetic algorithm," *Scientia Iranica*, vol. 18, no. 6, pp. 1600–1608, 2011.

[7] S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao *et al.*, "Interpretability of deep learning models: a survey of results," in *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable*

computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI). IEEE, 2017, pp. 1–6.

[8] Q.-s. Zhang and S.-C. Zhu, "Visual interpretability for deep learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 27–39, 2018.

[9] T. Linjordet and K. Balog, "Impact of training dataset size on neural answer selection models," in *European Conference on Information Retrieval*. Springer, 2019, pp. 828–835.

[10] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *arXiv preprint arXiv:1712.00409*, 2017.

[11] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Computers and electronics in agriculture*, vol. 153, pp. 46–53, 2018.

[12] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, "Improving defect prediction with deep forest," *Information and software technology*, vol. 114, pp. 204–216, 2019.

[13] R. Su, X. Liu, L. Wei, and Q. Zou, "Deep-resp-forest: A deep forest model to predict anti-cancer drug response," *Methods (San Diego, Calif.)*, vol. 166, pp. 91–102, 2019.

[14] X. Zeng, S. Zhu, Y. Hou, P. Zhang, L. Li, J. Li, L. F. Huang, S. J. Lewis, R. Nussinov, and F. Cheng, "Network-based prediction of drug–target interactions using an arbitrary-order proximity embedded deep forest," *Bioinformatics*, vol. 36, no. 9, pp. 2805–2812, 2020.

[15] L. Mou, S. Mao, H. Xie, and Y. Chen, "Structured behaviour prediction of on-road vehicles via deep forest," *Electronics letters*, vol. 55, no. 8, pp. 452–455, 2019.

[16] M. A. Ahmad, C. Eckert, and A. Teredesai, "Interpretable machine learning in healthcare," in *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, 2018, pp. 559–560.

[17] P. Hajek, "Interpretable fuzzy rule-based systems for detecting financial statement fraud," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2019, pp. 425–436.

[18] G. Zhu, Q. Hu, R. Gu, C. Yuan, and Y. Huang, "Forestlayer: Efficient training of deep forests on distributed task-parallel platforms," *Journal of parallel and distributed computing*, vol. 132, pp. 113–126, 2019.

[19] M. Zhou, X. Zeng, and A. Chen, "Deep forest hashing for image retrieval," *Pattern recognition*, vol. 95, pp. 114–127, 2019.

[20] J. Gao, K. Liu, B. Wang, D. Wang, and Q. Hong, "An improved deep forest for alleviating the data imbalance problem," *Soft Computing*, vol. 25, no. 3, pp. 2085–2101, 2021.

[21] L. Yang, X.-Z. Wu, Y. Jiang, and Z.-H. Zhou, "Multi-label learning with deep forest," 2019.

[22] L. Utkin, A. Konstantinov, A. Meldo, M. Ryabinin, and V. Chukanov, "A deep forest improvement by using weighted schemes," in *2019 24th Conference of Open Innovations Association (FRUCT)*. IEEE, 2019, pp. 451–456.

[23] L. V. Utkin and M. A. Ryabinin, "A siamese deep forest," *Knowledge-based systems*, vol. 139, pp. 13–22, 2018.

[24] L. V. Utkin, "An imprecise deep forest for classification," *Expert systems with applications*, vol. 141, p. 112978, 2020.

[25] S. Marsland, *Machine learning: an algorithmic perspective*. CRC press, 2015.

[26] O. Sagi and L. Rokach, “Ensemble learning: A survey,” vol. 8, no. 4, p. n/a, 2018.

[27] A. Mikołajczyk and M. Grochowski, “Data augmentation for improving deep learning in image classification problem,” in *2018 international interdisciplinary PhD workshop (IIPhDW).* IEEE, 2018, pp. 117–122.

[28] L. V. Utkin, “An imprecise deep forest for classification,” *Expert systems with applications*, vol. 141, p. 112978, 2020.

[29] X. Cao, L. Wen, Y. Ge, J. Zhao, and L. Jiao, “Rotation-based deep forest for hyperspectral imagery classification,” *IEEE geoscience and remote sensing letters*, vol. 16, no. 7, pp. 1105–1109, 2019.