FiberNet: A Compact and Efficient Convolutional Neural Network Model for Image Classification

Verner Rafael Ferreira[1], Anne Magaly de Paula Canuto[1]

[1]Departamento de Informática e Matemática Aplicada – DIMAp

Universidade Federal do Rio Grande do Norte (UFRN) – Natal – Brazil.

vrferreira@uneb.br, anne@dimap.ufrn.br

Abstract. *Creating a convolutional neural network (CNN) with a minimal number of trainable parameters can offers benefits across diverse application domains. In our paper, presents FiberNet, an efficient CNN model with reduced trainable parameters that guarantees both high accuracy and swift inference. Through empirical analysis confirming its efficacy, the FiberNet model achieved a remarkable 96.25% accuracy on the sisal dataset, 74.90% on the CIFAR10 dataset, and incorporates a total of 754,345 trainable parameters.*

1 Introduction

Neural network models have multifarious applications. Notably, they play a pivotal role in situations where algorithms need to function on memory-constrained hardware like smartphones or tablets. In such cases, models with minimal trainable parameters, often termed "mobile" models, are favored. These neural networks become crucial, particularly when crafting applications involving confidential data, which can't be transmitted remotely via servers due to privacy constraints. Medical records, financial data, and personal information applications exemplify this. As technology evolves, these nimble models continue to reshape our approach to mobile and privacy-centric applications. Furthermore, these neural network models can provide real-time results even in a situation where you have no internet or phone signal [Sandler et.al. 2018] and [Howard A. et al, 2019].

For the CNN-based models, there are many examples of mobile mobiles, such as MobileNetV1 [Howard et.al. 2017], SqueezeNet [09] and ShuffleNetV2 [Ma et.al. 2018]. These networks were specifically designed to meet the demands of mobile devices, being able to perform complex tasks in real time without compromising device performance. SqueezeNet, for instance, contains approximately 1 million parameters, which is significantly less than other convolutional neural networks, such as VGG16 [Simonyan and Zisserman 2014], which contains more than 138 million parameters. Despite having fewer parameters, SqueezeNet is still capable of providing accurate results in different image classification and object detection tasks.

Aiming to contribute to this important field, this paper proposes a simple and efficient Convolutional Neural Network (CNN) model. This model, named FiberNet, can be applied mainly in the computer vision field and its main feature is to extract useful information from images and use it in the inference process. In order to do this in a robust way, the proposed model includes a specific layer in the CNN architecture in order to reduce the image input before the convolution layers. In this sense, the

convolutional process becomes less complex, leading to a reduction in its computational cost.

In order to evaluate the feasibility of the proposed method, an empirical analysis will be conducted. It is important to emphasize that the proposed model can be applied to any image classification application. In this paper, the empirical analysis is focused on two image classification datasets. The first one is an in-house dataset created for Agave Sisalana plant classification, which contains only two classes: the sisal plant and the sisal fiber. The second dataset is a well-known image dataset, CIFAR10, which consists of 60000 32x32 color images in 10 classes, with 6000 images per class.

This paper is divided into 6 sections, and it is organized as follows. Details about the description of different CNN-based models are presented in Section 2. In Section 3, we present the proposed method, FiberNet, while the experimental methodology is described in Section 4. In section 5, the experimental results are presented and analyzed. Finally, in section 6, we present the final remarks of this paper and possibilities for future works

2 Groundwork for CNN Development

The general CNN algorithm is composed of two main parts: the first part is made of two layers, the convolution and pooling layers. The first one applies a convolutional function to create a feature map that summarizes the presence of detected features in the input image. The output of the convolutional layers is presented to the pooling layer; the primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs [Gholamalinezhad and Khosravi 2020]. These maps are created when input is passed through one or more convolution layers and pooling layers that use different kernel sizes [Agrawal and Mittal, 2020]. In the second part of a CNN architecture, a fully connected or dense network is used. The dense component of the network plays a crucial role in interpreting the values acquired during the initial phase. This segment can encompass one or several hidden layers with varying numbers of neurons, contributing to the networks overall complexity and capacity for learning intricate patterns.

2.1 Variations in CNN architectures

Several CNN-based models have been developed in the literature. AlexNet [Krizhevsky et.al. 2012], for instance, uses multiple GPUs, while VGG [Simonyan and Zisserman 2014] has up to 19 layers deep. Furthermore, GoogleNet [Szegedy et.al. 2015] has innovated with Naive Inception modules. On the other hand, ResNet [Kaiming et.al. 2016] and DenseNet [Hinton et.al 2012] use Skip Connections and residual blocks to solve the vanishing gradient problem [Sepp 1998], while SqueezeNet [Iandola et.al 2016] introduces the squeeze-and-excite pattern in fire modules. MobileNetV2 [Sandler et.al. 2018] and MobileNetV3 [Howard et.al. 2019] use Inverted Residual Blocks, while ShuffleNetV2 [Ma et.al. 2018] uses the Channel Algorithm Shuffle. Finally, EfficientNetV1 [Tan and Le 2019] improves performance through scalability of width, depth, or resolution, and EfficientNetV2 [Tan and Le 2021] includes a new convolution

block called Fused-MBConv and it uses progressive learning and Neural Architecture Search.

2.2 Related Works

In the field of deep learning, there are several mobile algorithms that have been developed. These models usually contain few trainable parameters and they can be considered as fast and efficient models. These characteristics of a mobile model can be achieved using specific methodologies as the ones applied in algorithms developed for mobile devices such as the ones proposed by [Tan et.al. 2019], [Huang 2018], [Chen et.al 2019], among others.

There are also those that are aimed at reducing trainable parameters through down sampling strategies such as those that apply pooling strategies and examples of these models can be found in He et al. [Kaiming et.al. 2015], [Zeiler and Fergus 2014], and [Graham 2014]. Additionally, there are models that use stride convolution strategies such as the ones proposed by [Yu and Koltun 2015] and [Zagoruyko and Komodakis 2016]. Models can also use transposed convolution strategies to reduce the number of trainable parameters, such as Zeiler and Fergus [Zeiler, Taylor and Fergus 2011] and [Dumoulin, and Visin 2016]. Finally, there are CNN-based models that use dilated convolution as in [Yu, Koltun and Funkhouser 2017].

3 The Proposed Model

The main contribution of the proposed model is the inclusion of a new layer, named Defiber layer, and it is included in the proposed CNN model, as presented in Figure 1. Its main objective is to reduce the input size before each convolution layer. In this sense, the input images are less complex, leading to a reduction in the number of trainable network parameters without deteriorating its accuracy. Figure 1 presents the general architecture of the proposed model FiberNet [1].

In order to describe the FiberNet functioning, the input image enters the first Defiber layer (DEFIBER * A). In this case, the Defiber layer decreases the image size of the input image by removing the indices lines and columns whose values are divisible by a threshold (t). The currently used threshold (t) is three because it was the value that, through our empirical analyses, demonstrated an accuracy equivalent to the other thresholds while using a smaller number of repetitions per Defiber layer, as described in Table 01 with rounded values and without using the 10-fold methodology.

This reduction is made (n) times, in which (n) is defined as described in the next paragraph. Then, the reduced image is sent to the convolutional layer. The result of the convolutional layer is then sent to the next Defiber layer (DEFIBER * B). This process is performed throughout the 12 layers of the model.

Moreover, Algorithm 1 outlines the key steps of the Defiber layer. As it can be observed in Figure 1, the Defiber layers have one parameter, which is the number of

[1] FiberNet code:
https://colab.research.google.com/drive/15-IjOcJbJudxx_RofJrVqtHHtyHbps35?usp=sharing

times it has to be repeated (n). In Figure 1, this parameter is represented by the variables A, B, C, D, E and F, for all six Defiber layers, respectively.

During the convolution process, these variables are automatically defined in the Defiber layer after an analysis of the image size by the Calculate Defiber Repeats (CDR) function, as described in Algorithm 2. This function consists of six nested loops, where inside the last loop, there is a simulation of the convolution part of our model's forward pass. In this simulation, there are copies of 12 layers: 6 Defiber layers and 6 convolution layers with similar functionality to the original layers.

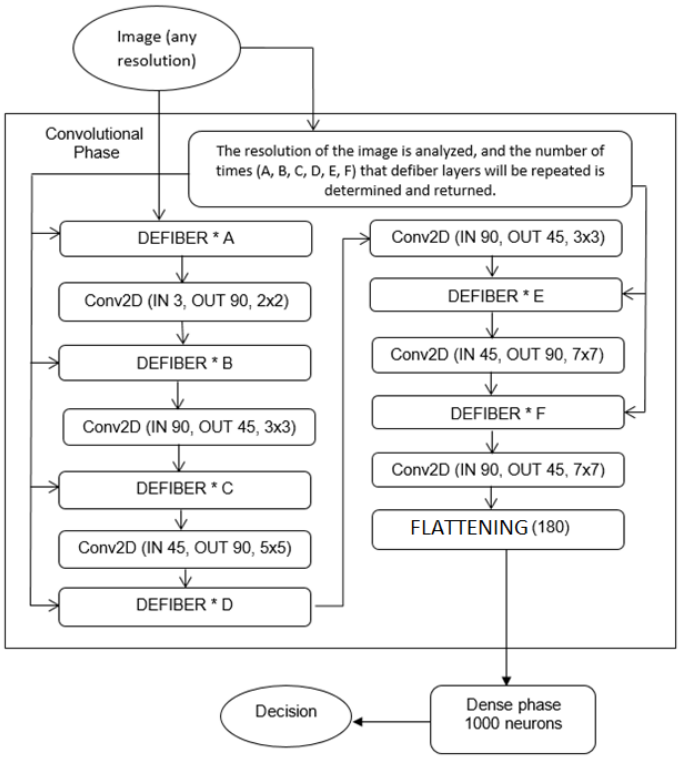

Fig. 1. FiberNet Architecture.

Table 1: Results of the empirical analysis of thresholds in the Defiber layer

Defiber threshold	ACC Cifar10	Number of repetitions	Parameters
01	---	00	Error: Excessively reduced.
02	0,74	01	754,345
03	0,75	01	754,345
04	0,74	02	754,345
05	0,72	03	754,345
06	0,75	03	754,345
07	0,75	04	754,345
08	0,75	04	754,345
09	0,72	04	754,345
10	0,73	05	754,345

This simulation is necessary for the model to predict, based on the image size, how many times the Defiber layers need to be repeated throughout the convolution

process for a given image size. It is based on this principle that our model can convolve images of any size. The main difference between the original forward flow and the simulation flow is in the type of return from each layer. In the original flow, we pass tensors between the layers, while in the simulation flow, we pass an integer variable representing the size of one of the dimensions of the input tensor between the simulation layers.

In the simulated flow, each simulated Defiber layer returns an integer representing the final size of the list where the indices of the non-divisible axes by the threshold (t), described as NewList (Algorithm 3). In turn, the simulated convolution layers also return an integer representing the size of the output calculated using the following formula: output_size = (input_size + (padding_size * 2) - kernel_size) // stride_size + 1.

Returning to the explanation about the loops, each loop has a variable (A, B, C, D, E, F) that iterates over a list containing values from 0 to 4. In the inner part of the last loop, these variables are passed as parameters to the simulated Defiber layers. The first DEFIBER layer receives A, the second DEFIBER layer receives B, and so on until the last DEFIBER layer receives F, as described in Algorithm 2. The flow continues with the Defiber layers being alternated with the simulations of the convolution layers (Defiber->Conv->Defiber->Conv...). After the last simulated convolution layer, if the obtained value is equal to 2, the function returns the combination of values of variables A, B, C, D, E, and F to the original flow.

The goal of this simulated flow is to determine the final output size after passing through the convolution flow. With the identified size, it serves as a basis for the model to find the correct number of times the Defiber layer in the original flow should be repeated.

Algorithm 1: Defiber layer.

```
01: procedure Defiber
02: INPUT tensor X
03: INPUT int var NumRepeatTimes
04: NEW int LIST NewList
05: FOR var i equals zero until size of NumRepeatTimes DO
06:   FOR var j equals zero until size of last DIM in X DO
07:     IF var j is not divisible by threshold THEN
          ADD var j in NewList
08:     END IF
09: END FOR
10: //X resized on the H and W axes
    RETURN X[:,:,NewList,:][:,:,:,NewList]
11: END procedure
```

Algorithm 2: Calculate Defiber Repeats function.

```
01: Procedure Calculate Defiber Repeats (CDR)
02: INPUT tensor X
03: NEW int LIST NewList filled as [0,1,2,3,4]
04: NEW int VAR OUT filled as value of last dim size in
```

```
    X
05: FOR VAR A equals zero until size of NewList DO:
06:   FOR VAR B equals zero until size of NewList DO:
        (...) repeat more 4 times to C, D, E and F
07: // In the loop F, we start the simulation of the forward pass.
      Both var OUT and var A are passed as
      parameters into the DefiberSimulation function,
      and returned as var OUT'.
08:     OUT' is passed as parameter into conv layer
        simulated, the output size is calculated and
        returned as var OUT''
          (...) Repeat this procedure throughout all
              simulated layers and at the end returned
              var OUT'''
09:     IF OUT''' equals 2 THEN
10:       RETURN the values of A,B,C,D,E,F to be set in
          the Defiber layer on main flow.
11:     END IF
12: END FOR
13: END procedure
```

Algorithm 3: Simulation of the Defiber layer.

```
01: procedure DefiberSimulation
02: INPUT int var LastDimSize
03: INPUT int var NumRepeatTimes
04: NEW int LIST NewList
05: FOR var i equals zero until size of NumRepeatTimes DO
06:   FOR var j equals zero until LastDimSize DO
07:     IF var j is not divisible by threshold (t) THEN
          ADD var j in NewList
08:     END IF
09: END FOR
10: RETURN size of X
11: END procedure
```

Describing the details of the convolutional layers, we have a minimal number of input and output channels configured to maintain the model accuracy. We have a sequence of kernels values (2x2, 3x3, 5x5, ... 7x7) configured based on our empirical analysis, padding set to zero and stride set to one in all layers. In addition, we use batch normalization (batch norm) to improve the learning speed and in order to avoid overfitting. Finally, we apply ReLU as an activation function in all convolutional layers. Additionally, a dense network with 1000 neurons is used as the default configuration.

4 Experimental analysis

In order to implement this empirical analysis, the Google Colab online development environment (https://colab.research.google.com) is used. This online platform provided by Google offers a programming environment compatible with python (used in deep learning) and also a quota of memory space and GPU processing. In addition, the Pytorch framework (https://pytorch.org/) is used to build the CNN architecture.

4.1 Used Datasets

The Sisal plant image dataset used in this empirical analysis was built by the authors of this paper (in-house dataset). The Sisal images were photographed on a smartphone camera and have files divided into two classes: fiber (plant by-product) and plant (original Sisal image). The images are configured in the size of 200x200 pixels as in the example in Figure 2. The Sisal dataset is composed of 826 images divided into two classes [plant] and [fiber], having 413 images for each class.

All images in each class were randomly distributed in the three used folders: training, validation and testing. The division into these three folders obeys the ratio 70/15/15 for training, validation and testing, respectively. This division is done in a stratified way in which each folder contains 50% of plant images and 50% of fiber images. This procedure is repeated, leading to 10 different results (dataset 01, dataset 02, dataset 03, ..., dataset 10). The results presented in the next section represent the average values over these 10 divisions.

The other image dataset is CIFAR10. This dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. It is composed of several different images, such as: airplane, ships, car, cats, dogs, among others. We decided to include tests with a different dataset as it presents a greater challenge to the analyzed models due to the complexity associated with the varying number of classes and image resolutions. Images with different resolutions allow us to observe the behavior of the models in scenarios with both more information (200x200) and less information (32x32). Additionally, it is a classical image dataset and it is well-known for several researchers in the area. Finally, the same methodology used with the sisal image dataset will also be applied with this one.

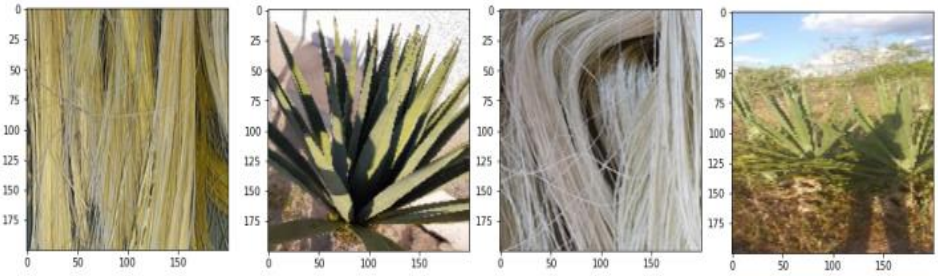

Fig. 2. Examples of sisal fiber and plant images

4.2 Methods and Materials

The proposed method and nine other CNN architectures (GoogleNet [Szegedy et.al., 2015], ResNet18 [Kaiming et.al. 2016], DenseNet121 [Huang et.al. 2017], SqueezeNet [Iandola et.al 2016], ShuffleNet [Ma et.al. 2018], MobileNetV2 [Sandler et.al. 2018], EfficientNetV1 [Tan and Le 2019], MobileNetV3 [Howard et.al. 2019] and EfficientNetV2 [Tan and Le 2021] will be evaluated on datasets described in Section 4.1. They were chosen for representing the state-of-the-art of CNNs currently. The evaluation criteria are based on David and Paul Teich's [Teich and Teich 2018] challenges for AI-based services, which include programmability, latency, accuracy, model size, and throughput. In this paper, we will use some of these criteria to assess the quality of the analyzed models.

Latency measures the response time of a CNN, while throughput is the maximum number of instances a model can process within a given time. To evaluate these criteria, we used the method described by [Geifman, 2020]. We assessed accuracy using the confusion matrix [Pearson 1904] to determine true positive, false positive, true negative, and false negative rates, and calculated accuracy using the formula ((TP+TN)/(TP+FP+TN+FN)). In order to represent the entire architecture of evaluated models, we assess the total number of trainable parameters (model size).

We also performed statistical analyses using the Friedman [Friedman 1937] and Nemenyi [Nemenyi 1963] post-hoc tests. We formulated null hypothesis h0 that all sample groups are equivalent and alternative hypothesis h1 that one or more sample groups come from different populations. Rejecting the null hypothesis with p-value $\leq \alpha$ (0.05) indicates that the alternative hypothesis is true. The Nemenyi test was then applied for pairwise analysis, with p-value ≥ 0.05 considered as similar.

In order to carry out the training phase, we use an online notebook available at Google Colab. The processing configuration at the time of carrying out the experiments was 10.81 GB GPU and 2.82 GB of RAM memory.

The hyper-parameter settings used in this paper are the following: batch- size is set to 4; optimal batch-size (used to measure the throughput and inference time only) is set to 64 (Maximum value before memory overflow); epochs are set to 50 (Chosen because it is sufficient for the models to demonstrate their quality); learning rate is set to 0.0001; loss function is set to cross-entropy loss and optimizer is set to Adam (LR: 0.0001). In our research, we are not using any type of data augmentation and all models were trained from scratch without using weight initialization.

5 The Experimental Outcomes

5.1 Results with the Sisal dataset

Table 2 displays the accuracy outcomes of the 10 analyzed methods. Furthermore, Table 3 showcases the throughput (TT) and inference time (TI) results, and Table 4 presents the trainable parameter outcomes.

Table 2. Accuracy result for the sisal dataset.

Models	Accuracy
SqueezeNet	98.96%
DenseNet	98.65%
ResNet	98.02%
FiberNet	96.25%
GoogleNet	95.94%
EfficientNetV2	94.79%
MobileNetV2	83.23%
ShuffleNet	81.25%
MobileNetV3	73.75%
EfficientNetB0 (V1)	69.48%

Table 3. Result of TT, TI for the sisal dataset.

	Models	TT	Models	TI

01	FiberNet	4464	FiberNet	3,89
02	MobileNetV3	2586	MobileNetV3	10,66
03	ShuffleNet	720	SqueezeNet	11,45
04	SqueezeNet	561	MobileNetV2	21,19
05	MobileNetV2	428	ShuffleNet	25,26
06	ResNet	406	EfficientNetV1	29,62
07	GoogleNet	330	ResNet	37,37
08	EfficientNetV1	314	GoogleNet	44,58
09	EfficientNetV2	299	EfficientNetV2	48,22
10	DenseNet	136	DenseNet	106,72

Table 4. Models and their respective parameters and file size

Models	Parameters	(MB)
FiberNet	754,345	2.9
SqueezeNet	1,248,424	5
ShuffleNet	2,278,604	9
MobileNetV3	2,542,856	9.7
MobileNetV2	3,504,872	14
EfficientNetV1	5,288,548	20
GoogleNet	6,624,904	25
DenseNet	8,534,408	31
ResNet	11,689,512	45
EfficientNetV2	22,103,832	84

In general, when analyzing a CNN architecture, it must have a tradeoff between performance (accuracy) and efficiency (TI, TT and trainable parameters). For instance, when comparing DenseNet and FiberNet, in terms of accuracy, DenseNet obtained an accuracy 2.4 percentage points (p.p.) higher than FiberNet. Nevertheless, Fibernet is much more efficient than DenseNet, being 96.35 p.p. faster in terms of TI and having 96.95 p.p. more instances (TT). Finally, Fibernet has 91.16 p.p. less parameters than DenseNet. In other words, Fibernet achieved slightly less performance than DenseNEt, but much better results in terms of efficiency (TT, TI and trainable parameters).

When comparing the CNN architectures with the best performance, it can be observed that FiberNet obtained the best trainable parameter results, followed by SqueezeNet (+39.57 p.p.), ShuffleNet (+66.89 p.p.), MobileNetV3 (+70.33 p.p.) and MobileNetV2 (+78.47 p.p.). However, we can also observe that the small number of trainable parameters delivered by FiberNet did not negatively impact its accuracy results. This is because all six models that obtained accuracy results close to 95% (SqueezeNet, DenseNet, ResNet, FiberNet, GoogleNet and EfficientNetV2) showed an average accuracy difference less than 3 p.p.

In the same perspective, it can be observed that there was a considerable reduction in the number of instances (throughput criterion - TT) processed by the other models, when compared to FiberNet: MobileNetV3 (-42.06 p.p.), SqueezeNet (-87.43 p.p.), MobileNetV2 (-90.41 p.p.) and ShuffleNet (-83.87 p.p.). The same observation occurs for inference time (IT) results in which there was a decrease in the processing speed delivered by FiberNet, when compared to MobileNetV3 (+63.5 p.p.), SqueezeNet (+66.02 p.p.), MobileNetV2 (+81.64 p.p.) and ShuffleNet (+84.6 p.p.).

Therefore, the results obtained in Tables 2, 3 and 4 demonstrated that the application of the Defiber layer associated with a small number of convolution layers did not negatively impact the accuracy results of FiberNet. In addition, the current configurations of the hyper-parameters used in FiberNet were sufficient to achieve an accuracy equivalent to the best CNN models. Unlike other CNN models that generally obtain learning gains from successive layers in depth, applying a large number of channels and using kernels of different sizes, our model was able to obtain learning from the approximation of areas of interest, by reducing gradual input.

In order to support the obtained results, we now present the results of the statistical tests with the sisal dataset. We begin by presenting the results for the Friedman test (Table 5). Since the p-value was less than 0.05, for all three criteria, we rejected the null hypothesis (h0) and applied the post-hoc test to determine, through a pairwise comparison, which pairs are more relevant. The post-hoc Nemenyi result for the accuracy criterion is presented in Table 6. In this table, we compare only the p-value results of the best result, that was the result of SqueezeNet, in relation to the other models.

According to the statistical analysis, the following CNN models: SqueezeNet, FiberNet, DenseNet, ResNet, GoogleNet and EfficientNetV2 can be considered equivalent, since all of them obtained a p-value ≥ 0.05. Concluding the results of the Sisal dataset, Table 7 presents the results of the post-hoc Nemenyi test for the throughput and inference time criteria.

For the inference rate and transfer rate criteria, the Nemenyi test showed us that we can also consider as equivalent the results of FiberNet, ShuffleNet, MobileNetV2, SqueezeNet and MobileNetV3 models. For these two criteria, we concluded that the best performance of FiberNet was probably due to the smaller amount of trainable parameters it has. Additionally, a quantitative advantage also promoted by the previous reduction of the input by the Defiber layer also had a positive effect in the FiberNet functioning.

5.2 Results with the CIFAR10 dataset

This section highlights the outcomes from the CIFAR10 dataset. Table 8 displays average accuracy values, while Table 9 presents TT and TI results. It's worth noting that the trainable parameters for models across both datasets remained equals, as shown in Table 04. From these tables and figure, it can be observed that no CNN model achieved accuracy equal to or higher than 95%. The best accuracy results were obtained by GoogleNet (75.9%), followed by FiberNet (-1 p.p.), ResNet (-12.5 p.p.), DenseNet (-14.5 p.p.), SqueezeNet (-17.5 p.p.) and EfficientNetV2 (-18.7 p.p.). A similar pattern of behavior to the sisal dataset was also obtained in this dataset.

Table 5. Friedman Test for accuracy, inference rate and throughput of the sisal dataset.

Criterion	P-value
Accuracy (ACC)	2.00448e-12
Inference Rate (TI)	1.78266e-15
Transfer rate (TT)	1.62807e-15

Table 6. Post-hoc Accuracy result of the sisal dataset.

Models	Post hoc
SqueezeNet	1,000
FiberNet	0.900
DenseNet	0.900
ResNet	0.900
GoogleNet	0.656
EfficientNetV2	0.633
MobileNetV2	0.001
ShuffleNet	0.001
MobileNetV3	0.001
EfficientNetV1	0.001

Table 7. Post-hoc results for TT and TI criteria of the sisal dataset

	Models	P-hoc TT	Models	P-hoc TI
01	FiberNet	1.00	FiberNet	1.00
02	ShuffleNet	0.900	SqueezeNet	0.900
03	MobileNetV3	0.900	MobileNetV3	0.900
04	SqueezeNet	0.449	MobileNetV2	0.449
05	MobileNetV2	0.090	ShuffleNet	0.090
06	ResNet	0.008	EfficientNetB0	0.008
07	GoogleNet	0.001	GoogleNet	0.001
08	DenseNet	0.001	DenseNet	0.001
09	EfficientNetB0	0.001	ResNet	0.001
10	EfficientNetV2	0.001	EfficientNetV2	0.001

In terms of efficiency measurements, keeping the same number of trainable parameters from the previous section, we noted fluctuations in the throughput (TT) values across all ten models. The highest TT achievement was by MobileNetV3, followed by ShuffleNet (-25.76 p.p.), SqueezeNet (-49.41 p.p.), FiberNet (-53.09 p.p.), ResNet (-53.68 p.p.), and MobileNetV2 (-63.64 p.p.). Similarly, there was variability in the inference time (IT) outcomes. SqueezeNet outperformed in this regard, followed by FiberNet (+18.56 p.p.), ResNet (+37.86 p.p.), MobileNetV3 (+57.84 p.p.), MobileNetV2 (+62.73 p.p.), and ShuffleNet (+69.13 p.p.), respectively.

Table 8. Accuracy result for the CIFAR10 dataset

Models	Accuracy
GoogleNet	75.9%
FiberNet	74.9%
ResNet	63.4%
DenseNet	61.4%
SqueezeNet	58.4%
EfficientNetV2	57.2%
MobileNetV3	44.3%
ShuffleNet	42.5%
MobileNetV2	40.3%
EfficientNetV1	36.6%

Table 9. TT and TI result for the CIFAR10 dataset.

	Models	TT	Models	TI
01	MobileNetV3	2550	SqueezeNet	4.30
02	ShuffleNet	1893	FiberNet	5.28

03	SqueezeNet	1290	ResNet	6.92
04	FiberNet	1196	MobileNetV3	10.20
05	ResNet	1181	MobileNetV2	11.54
06	MobileNetV2	927	ShuffleNet	13.93
07	GoogleNet	903	GoogleNet	16.23
08	EfficientNetV1	746	EfficientNetV1	16.73
09	DenseNet	314	DenseNet	41.41
10	EfficientNetV2	308	EfficientNetV2	43.35

Therefore, considering the results obtained with this dataset, we can conclude that the use of the Defiber layer associated with a reduced convolutional environment did not negatively influence the model prediction process, since FiberNet provided a promising accuracy result and competing efficiency results.

When comparing the results of both datasets, in terms of accuracy, we observed that the same six CNN models that provided good accuracy levels on the sisal dataset also achieved good accuracy levels in this dataset. In terms of the efficiency results, the proposed CNN model managed to be among the five best results, in both criteria.

In order to validate the results obtained on this dataset, the statistical analysis is done and Table 10 presents the results of the Friedman test. It is important to emphasize that the results of the trainable parameters are the same as the previous dataset and it was not included in this table.

Based on the result of the Friedman test, we rejected the null hypothesis (h0), for all three criteria. Then, the post-hoc Nemenyi tests are applied for accuracy (Table 11), TT and TI (Table 12).

By the results of the accuracy, TT and TI criteria, we can see that the proposed CNN model always remained among the top five methods. In addition, it can be always considered statistically equivalent to the best models, for all criteria. It shows that the proposed method managed to reduce the computational complexity of a CNN model without deteriorating its accuracy.

Table 10. Friedman results for accuracy, TT and TI

Criterion	P-value
Accuracy (ACC)	2.00295e-15
Inference rate (IT)	2.20275e-15
Transfer rate (TT)	2.01180e-15

Table 11. Post-hoc Accuracy result of the CIFAR10 dataset.

	Models	Accuracy
01	GoogleNet	1.00
02	FiberNet	0.900
03	ResNet	0.900
04	DenseNet	0.497
05	SqueezeNet	0.100
06	EfficientNetV2	0.014
07	MobileNetV3	0.001
08	MobileNetV2	0.001
09	ShuffleNet	0.001
10	EfficientNetB0	0.001

Table 12. Post-hoc results for TT and TI.

	Models	TT	Models	TI
01	MobileNetV3	1.00	SqueezeNet	1.00
02	SqueezeNet	0.900	FiberNet	0.900
03	ShuffleNet	0.900	ResNet	0.900
04	FiberNet	0.303	MobileNetV3	0.449
05	ResNet	0.160	MobileNetV2	0.090
06	MobileNetV2	0.008	ShuffleNet	0.008
07	GoogleNet	0.001	GoogleNet	0.001
08	EfficientNetB0	0.001	EfficientNetB0	0.001
09	DenseNet	0.001	DenseNet	0.001
10	EfficientNetV2	0.001	EfficientNetV2	0.001

6 Concluding Remarks

Introducing FiberNet, a compact and rapid CNN model characterized by a lean number of trainable parameters. Its primary objective lies in executing a Defiber layer that reduces image dimensions while preserving the embedded information. Initially devised for the classification of a specific plant, FiberNet's efficiency extends to broader domains. Demonstrating swift and precise image processing, FiberNet operates without the demand for extensive computational resources. An empirical assessment was executed to gauge the viability of this approach.

Within this analysis, FiberNet's implementation encompassed 754,345 trainable parameters, achieving a remarkable 96.25% accuracy in classifying sisal plants. Further evaluation was carried out using the widely recognized CIFAR10 image dataset. Impressively, FiberNet secured the second-highest accuracy, trailing only GoogleNet by a one percentage point. Notably, FiberNet outperforms GoogleNet in terms of efficiency, excelling in criteria such as TT, TI, and trainable parameters. This commendable efficiency is attributed to the incorporation of the Defiber layer, reducing input image dimensions before each convolutional layer.

Looking ahead, our research plans encompass the scrutiny of our methodology's impact on CNN models with diverse hyper-parameter configurations. Additionally, we intend to assess its performance across sisal image datasets of varying resolutions and examining FiberNet's performance sans the Defiber layer. We propose its replacement with pooling layers, facilitating a comparative analysis against FiberNet's current performance.

7 References

Agrawal, A. and Mittal N. (2020). “Using CNN for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy”. The Visual Computer, v. 36, no. 2, p. 405-412.

Chen, Y. et al (2019). “Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution”. In: Proceedings of the IEEE/CVF international conference on computer vision. p. 3435-3444.

Dumoulin, V. and Visin, F. (2016). “A guide to convolution arithmetic for deep learning”. arXiv preprint arXiv:1603.07285.

Friedman, M. (1937). “The use of ranks to avoid the assumption of normality implicit in the analysis of variance”. Journal of the American Statistical Association, v. 32, no. 200, p. 675-701.

Graham, B. (2014). “Fractional max-pooling”. arXiv preprint arXiv:1412.6071.

Geifman, A. (2020). “The Correct Way to Measure Inference Time of Deep Neural Networks”. Available at: "https://towardsdatascience.com/the -correct-way-to-measure-inference-time-of-deep-neural-networks-304a54e5187f" (Accessed on: March 22, 2021).

Gholamalinezhad, H. and Khosravi, H. (2020). “Pooling Methods in Deep Neural Networks, a Review”. arXiv preprint arXiv:2009.07485.

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

Howard, A. et al (2017). “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. arXiv preprint arXiv:1704.04861.

Howard, A. et al (2019). “Searching for mobilenetv3”. In: Proceedings of the IEEE/CVF international conference on computer vision. p. 1314-1324.

Huang, G., Liu, Z., Maaten, L.V.D. and Weinberger, K.Q. (2017). “Densely connected convolutional networks”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. p. 4700-4708.

Huang, G. et al (2018). “Condensenet: An efficient densenet using learned group convolutions”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. p. 2752-2761.

Iandola, F.N. et al (2016). “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size”. arXiv preprint arXiv:1602.07360.

Kaiming, H. et al (2015). “Spatial pyramid pooling in deep convolutional networks for visual recognition”. IEEE transactions on pattern analysis and machine intelligence, v. 37, n. 9, p. 1904-1916.

Kaiming, H. et.al. (2016). “Deep residual learning for image recognition”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. p. 770-778.

Krizhevsky. A., Sutskever, I. and Hinton, G.E. (2012). “Imagenet classification with deep convolutional neural networks”. Advances in neural information processing systems, v. 25, p. 1097-1105.

Ma, N., Zhang, X., Zheng, H., and Sun, J. (2018). “ShuffleNet v2: Practical guidelines for efficient cnn architecture design”. In: Proceedings of the European conference on computer vision (ECCV). p. 116-131.

Nemenyi, PB. (1963). “Distribution-free multiple comparisons”. Princeton University.

Pearson, K. (1904). “On the theory of contingency and its relation to association and normal correlation”. Drapers' Company Research Memoirs. Biometric series I: Dulau and Co.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L. (2018). “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. p. 4510-4520.

Sepp, H. (1998). “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, v. 6, no. 02, p. 107-116.

Simonyan, K. and Zisserman, A. (2014). “Very deep convolutional networks for large-scale image recognition”. arXiv preprint arXiv:1409.1556.

Szegedy, C. et al (2015). “Going deeper with convolutions”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. p. 1-9.

Tan, M., and Le, Q., (2019). “EfficientNet: Rethinking model scaling for convolutional neural networks”. In: International Conference on Machine Learning. PMLR, p. 6105-6114.

Tan, M. et al (2019). Mnasnet: “Platform-aware neural architecture search for mobile”. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. p. 2820-2828.

Tan, M., and Le, Q., (2021). “Efficientnetv2: Smaller models and faster training”. In: International Conference on Machine Learning. PMLR, p. 10096-10106.

Teich, D.A. and Teich, P.R. (2018). PLASTER: “A Framework for Deep Learning Performance”. Tech. rep. TIRIAS Research.

Yu, F., Koltun, V. (2015). “Multi-scale context aggregation by dilated convolutions”. arXiv preprint arXiv:1511.07122.

Yu, F., Koltun, V., Funkhouser, T (2017). “Dilated residual networks”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. p. 472-480.

Zagoruyko, S., Komodakis, N. (2016). “Wide residual networks”. arXiv preprint arXiv:1605.07146.

Zeiler, M.D., Taylor, G.W., Fergus, R. (2011). “Adaptive deconvolutional networks for mid and high-level feature learning”. In: 2011 international conference on computer vision. IEEE, p. 2018-2025.

Zeiler, M.D. and Fergus, R. (2014). “Visualizing and understanding convolutional networks”. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13. Springer International Publishing, p. 818-833.