Potencializando a classificação de imagens com eficiência e robustez através da FiberNet.

Verner Rafael Ferreira
ORCID: *0009-0000-4066-5750*
Departamento de Informática e Matemática Aplicada - DIMAp
Universidade Federal do Rio Grande do Norte - UFRN
Natal, Brasil
vrferreira@gmail.com

Anne Magaly de Paula Canuto
ORCID: *0000-0002-3684-3814*
Departamento de Informática e Matemática Aplicada - DIMAp
Universidade Federal do Rio Grande do Norte - UFRN
Natal, Brasil
anne@dimap.ufrn.br

***Resumo*— Um modelo baseado em CNN que utiliza um pequeno número de parâmetros treináveis pode trazer vários benefícios importantes para diferentes domínios de aplicação. Entre eles, podemos mencionar o processo de treinamento mais rápido, o que significa que sua implementação pode ser feita de forma mais eficiente e rápida. Além deste, nós podemos citar também a redução significativa na probabilidade de overfitting e a redução no consumo dos recursos computacionais necessários para a execução, o que pode ser muito útil em sistemas embarcados e outros dispositivos com hardware limitado. Portanto, neste artigo, nós propomos o modelo FiberNet, que é uma rede neural convolucional (CNN) simples e eficiente que produz um pequeno número de parâmetros treináveis e alta velocidade de inferência. O objetivo principal é fornecer uma alternativa eficiente para aplicações móveis, bem como aplicações que requerem modelos compactos. Para avaliar a eficácia da FiberNet, nós realizamos um estudo empírico comparativo entre os resultados do nosso modelo e o resultado de outros modelos que são considerados como o estado da arte. Essa comparação foi realizada em dois conjuntos de dados de imagens diferentes, um dos quais é o conhecido conjunto de dados CIFAR10. Como resultado, o modelo FiberNet obteve uma precisão de 96,25% no conjunto de dados Sisal, 74,90% no conjunto de dados CIFAR10 e foi o primeiro em total de parâmetros treináveis com 754.345 (39,57% menor em relação ao segundo colocado).**

Palavras-chave*— *rede neural convolucional, classificação de imagem, deep learning, Agave Sisalana, Sisal.

I. INTRODUÇÃO

Ao longo dos anos, uma variedade de modelos de redes neurais tem sido desenvolvida para atender a diferentes propósitos e aplicações. Entre os inúmeros modelos já criados, surgem cenários em que esses algoritmos precisam ser executados em hardwares com limitada capacidade de memória, como smartphones ou tablets.

Nesses casos, geralmente optamos por modelos que possuem um pequeno número de parâmetros treináveis e esse tipo de modelo é tradicionalmente conhecido como modelos "*mobile*", "*tiny*" ou "*lightweight*". Esse tipo de rede neural pode ser especialmente importante quando desejamos desenvolver um aplicativo, por exemplo, que trabalhe com dados confidenciais que não podem ser compartilhados remotamente por meio de um servidor.

Informações como dados médicos, dados financeiros ou que envolvam informações pessoais. Além disso, esses modelos de rede neural podem fornecer resultados em tempo real mesmo em situações em que não há conexão com a internet ou sinal de telefone [16] [3].

Para os algoritmos baseados em CNN, existem muitos exemplos de modelos móveis, como a MobileNetV1 [11], SqueezeNet [09] e ShuffleNetV2 [19]. Essas redes foram especificamente projetadas para atender às demandas de dispositivos móveis, sendo capazes de realizar tarefas complexas em tempo real sem comprometer o desempenho do dispositivo.

A SqueezeNet, por exemplo, contém um pouco mais de 1 milhão de parâmetros, o que é significativamente menor do que outras redes neurais convolucionais, como o VGG16 [13], que possui mais de 138 milhões de parâmetros. Apesar de ter menos parâmetros, a SqueezeNet ainda é capaz de fornecer resultados precisos em diferentes tarefas de classificação de imagem e detecção de objetos.

Com o objetivo de contribuir para essa área importante, este artigo propõe um modelo de Rede Neural Convolucional (CNN) simples e eficiente. Esse modelo, chamado FiberNet, pode ser aplicado principalmente no campo da visão computacional e sua principal característica é extrair informações úteis de imagens e utilizá-las no processo de inferência.

Para fazer isso de maneira robusta, o modelo proposto inclui uma camada específica na arquitetura da CNN para reduzir a entrada da imagem antes das camadas de convolução. Nesse sentido, o processo de convolução se torna menos complexo, levando a uma redução em seu custo computacional.

Para avaliar a viabilidade do método proposto, realizou-se uma minuciosa análise empírica neste estudo, concentrando-se em dois conjuntos de dados para classificação de imagens. O primeiro conjunto é uma base interna especialmente criada para classificar a planta Agave Sisalana, com duas classes distintas: a planta sisal e a fibra sisal. Já o segundo conjunto de dados utilizado é o conhecido CIFAR10, composto por 60.000 imagens coloridas de 32x32 pixels, distribuídas em 10 classes, com 6.000 imagens por classe. Essa abordagem nos permite obter insights valiosos sobre a eficácia do método em diferentes cenários e reforça a robustez de nossa proposta.

Nós apresentamos todo o desenvolvimento de nossa pesquisa através deste artigo que está dividido em 06 seções. Na seção 02 nós apresentamos toda a conceituação sobre o que é uma CNN, suas principais características e aspectos relacionados à sua arquitetura para a realização da tarefa de classificação de imagens. Descrevemos também

os detalhes concernentes à subdivisão interna que uma CNN possui. Bem como essas partes contribuem para que as CNNs possam alcançar sucesso na realização de sua tarefa.

Além disso, é importante destacar que essas mesmas partes, quando configuradas ou modificadas adequadamente, possibilitam que o modelo seja classificado como mobile. Os modelos de CNN classificados como mobile são caracterizados por possuírem uma quantidade reduzida de parâmetros treináveis e apresentarem maior agilidade em relação aos modelos de CNN tradicionais.

É possível alcançar esse novo tipo de CNN através de ajustes ou modificações na sua arquitetura. Desde a inclusão de novas camadas de convolução, permitido que o modelo alcance novo patamares de profundidade, com combinação de filtros que permitam o redimensionamento da imagem, até mesmo à inclusão de novos tipos de camadas e/ou modificação nas camadas já existentes.

Sendo diretamente vinculados a esses conceitos, nós apresentamos também, na seção 03, a descrição detalhada da nossa proposta. Nosso modelo, denominado de FiberNet, é um algoritmo que possui como proposta inovadora a inclusão de uma nova camada, denominada Defiber, que é inserida na arquitetura tradicional de uma CNN mais especificamente na seção conhecida como fase de convolução. Essa camada tem por finalidade permitir que nosso modelo mantenha um baixo número de parâmetros treináveis, por meio de um processo de redução do tamanho da imagem de entrada (input), sem que para isso haja perda considerável da sua capacidade de inferência.

Nós apresentamos também, na subseção 3A, uma relação de modelos de CNN que se notabilizaram pela sua capacidade inovadora de lidar com o desafio de classificação de imagens e são conhecidas atualmente como o estado da arte nesta área. Além disso, nós apresentamos também, na subseção 3B, três trabalhos recentes que lidam com o problema da classificação de imagens a partir de um modelo leve, com baixa quantidade de parâmetros treináveis e que conseguem, dentro do cenário apresentado, um determinado percentual de acurácia.

Os trabalhos selecionados foram criteriosamente escolhidos devido às suas propostas inovadoras, relevância recente e notável contribuição para o aprimoramento das arquiteturas de CNN, permitindo que esses modelos apresentem um número reduzido de parâmetros treináveis. Dessa forma, são considerados exemplos de arquiteturas leves (lightweight) que proporcionam um desempenho eficiente.

Continuando a descrição das seções do artigo, na seção 04 nós apresentamos a descrição da metodologia adotada através do qual nós realizamos uma investigação comparativa entre os resultados obtidos pelo nosso modelo, a FiberNet, no desafio de classificar imagens da base do Sisal e da base CIFAR10, contra os resultados obtidos pelos modelos considerados como o estado da arte na área do aprendizado profundo (deep learning). Esses resultados são relevantes, pois demonstra, neste cenário, a validade do modelo proposto. Os resultados alcançados são detalhados na seção 05 e, por fim, na seção 6, apresentamos as considerações finais deste artigo e possibilidades para trabalhos futuros.

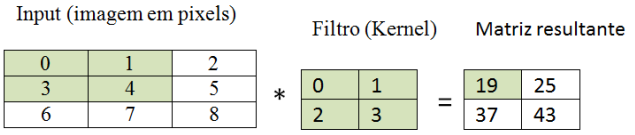
II. Rede Neural Convolucional

As CNNs (Convolutional Neural Networks) são um tipo de rede neural especialmente eficaz para tarefas que envolvem o processamento de imagens. Esses modelos têm a capacidade de aprender características de forma automática a partir das imagens, permitindo que sejam usados em diversas aplicações, como detecção de objetos, reconhecimento facial e classificação de imagens. Sua arquitetura única, com camadas de convolução e pooling, permite a extração eficiente de características relevantes das imagens, tornando-as uma escolha valiosa para tarefas de visão computacional.

A arquitetura geral de uma CNN é composta por duas partes principais: a primeira parte inclui as camadas de convolução e de pooling, enquanto a segunda parte inclui as camadas densas ou totalmente conectadas.

As camadas de convolução são responsáveis por aprender características nas imagens. Elas fazem isso aplicando uma função matemática às entradas, que é chamada de filtro. O filtro é um pequeno vetor de números que é usado para identificar padrões nas imagens (como ilustrado na figura 1). As camadas de convolução são aplicadas repetidamente, de modo que cada camada aprende características mais complexas do que a camada anterior.

Fig. 1. Ilustrção do processo de aplicação do filtro à matriz de entrada.



Nessa ilustração, temos uma matriz (tensor) de pixels 3x3 representando os valores de 0 a 8. A aplicação de um filtro (kernel) de 2x2 inicia-se posicionando (imagine como uma sobreposição) a tela do filtro no canto superior esquerdo da matriz de pixels e realizando a multiplicação elemento por elemento, somando os resultados para obter um valor central na nova matriz resultante.

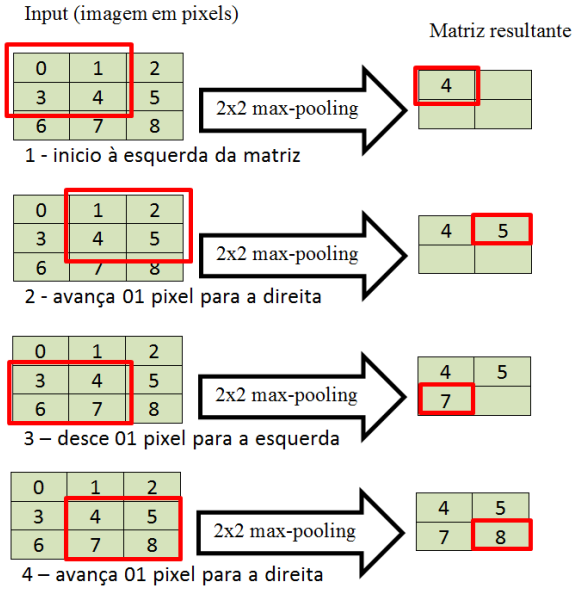
Como no exemplo apresentado os quatro primeiros valores da matriz de pixels são multiplicados pelos valores do filtro na ordem de posicionamento: (0*0) + (1*1) + (2*3) + (3*4) cujo resultado é 19. Em um segundo momento o filtro avança uma casa para a direita e a partir do novo posicionamento temos os cálculos de: (0*1) + (1*2) + (2*4) + (3*5) que resulta no valor 25, segundo valor da matriz resultante e assim por diante.

O filtro percorre toda a matriz de pixels, calculando os valores e avançando um pixel por vez ou conforme o valor do avanço (stride) estiver definido na sua rede. Ao concluir a primeira linha de pixels o filtro é reposicionado novamente no início à esquerda, uma linha de pixel abaixo, e o processo é continuado. As parametrizações de filtro e avanço garantem que as camadas de convolução possam percorrer toda a área de informação da imagem de maneira minuciosa (capturando mais detalhes pixel a pixel) ou de uma maneira mais abrangente (reduzindo significativamente a matriz resultante ao final)

A matriz final resultante é formada pelos valores obtidos em cada passo da convolução. Esse procedimento é essencial para extrair características e realizar operações em matrizes de pixels, sendo amplamente utilizado em processamento de imagens e redes neurais convolucionais.

Já as camadas de pooling [10] são responsáveis por reduzir o tamanho dos mapas de características criados pelas camadas de convolução. Isso é feito combinando grupos de pixels adjacentes em um único pixel. As camadas de pooling ajudam a reduzir a quantidade de dados que precisam ser processados pelas camadas densas, o que pode melhorar a eficiência da rede. Conforme ilustramos na figura 2.

Fig. 2. Exemplo da aplicação da camada de maxpooling.



Neste exemplo, consideramos uma matriz de 3x3 com valores de 0 a 8. Ao aplicarmos maxpooling com uma região 2x2, dividimos a matriz em quadrados de 2x2 e selecionamos o valor máximo de cada um. O processo resulta em uma nova matriz de 2x2, onde cada elemento representa o valor máximo encontrado em sua respectiva região na matriz original.

No exemplo em específico, utilizando à metodologia maxpooling, a camada de pooling reduz a quantidade de dados a serem processados preservando o pixel com maior valor de uma determinada região da matriz, considerando-a como sendo o valor de maior relevância. Esse, dentre outras, é um exemplo de metodologia que pode ser utilizada durante a fase de convolução para reduzir o tamanho da entrada (input).

As camadas densas ou totalmente conectadas têm como função principal a classificação das imagens. Nessa etapa, uma matriz de neurônios é interconectada, permitindo que cada neurônio receba os valores dos mapas de características provenientes das camadas de convolução e pooling. A saída de cada neurônio é um valor que representa a probabilidade de a imagem pertencer a uma classe específica.

Essas camadas são essenciais para a tarefa de classificação e são compostas por uma matriz de neurônios interconectados, cada neurônio recebe como entrada os valores dos mapas de características extraídos pelas camadas de convolução e pooling. Essa abstração das características permite que a rede interprete informações relevantes para a tarefa de classificação e produza uma saída, representando a probabilidade de a imagem pertencer a uma classe específica.

Essa interpretação dos dados pelas camadas densas é especialmente relevante nas redes neurais do tipo mobile ou tiny, projetadas para dispositivos com recursos computacionais limitados. Nesse contexto, as técnicas aplicadas nessas redes buscam aperfeiçoar o tamanho e a velocidade de processamento, tornando-as mais adequadas para execução em dispositivos como smartphones e dispositivos embarcados.

As redes neurais do tipo mobile ou tiny são variações das CNNs, especialmente projetadas para operar em dispositivos com recursos computacionais limitados, como smartphones e dispositivos embarcados. Para alcançar maior eficiência em termos de tamanho e velocidade de processamento, essas redes aplicam diversas técnicas avançadas, como o uso de camadas mais leves, como convoluções separáveis, que dividem as operações convolucionais em duas etapas, tornando a rede mais leve e reduzindo a quantidade de parâmetros treináveis.

Essas técnicas permitem que essas redes operem eficientemente em dispositivos com recursos limitados, mantendo um bom equilíbrio entre precisão e desempenho, principalmente ao controlar a quantidade de parâmetros treináveis resultante da aplicação dessas técnicas. Na próxima seção nós abordaremos a descrição de alguns modelos que se notabilizaram por utilizarem técnicas inovadoras neste sentido.

A quantidade de parâmetros treináveis de uma rede neural refere-se à quantidade de pesos e viés (bias) que a rede precisa aprender durante o processo de treinamento para ajustar os dados de entrada e gerar as saídas corretas. Esses parâmetros são fundamentais para o funcionamento da rede, pois eles determinam como as informações são processadas em cada camada e influenciam diretamente na precisão e desempenho do modelo.

Nesse contexto, a proposta que apresentamos neste artigo se destaca como uma solução inovadora, introduzindo um novo tipo de camada responsável pela redução do input antes de cada camada de convolução. Considerando que o processo de convolução, incluindo a quantidade de camadas envolvidas, demanda certo tempo de processamento, buscamos através de nossa metodologia reduzir esse tempo, diminuindo a quantidade de camadas e, consequentemente, o total de parâmetros treináveis.

Essa redução de parâmetros é um dos fatores que torna nosso modelo rápido no processo de inferência das imagens, sem comprometer a precisão das classificações. Com essa abordagem, nosso modelo se apresenta como uma opção para aplicações em dispositivos com recursos computacionais limitados.

A. As diferentes arquiteturas baseadas em CNN.

As CNNs são algoritmos complexos e atualmente existem muitas versões. Nesta seção, nós apresentamos, de maneira resumida, alguns dos principais algoritmos que são considerados como o estado da arte nesta área.

AlexNet [4]: É o modelo vencedor do ILSVRC com 85% de precisão e inovou ao usar múltiplas GPUs em seu processamento. Este modelo possui cinco camadas de convolução que utilizam kernels de (5x5) e (3x3) com função de ativação ReLU [40], três camadas de maxpooling com kernel de (3x3) e Dropout [35].

VGG [13]: A VGG se destaca por sua profundidade, suportando até 19 camadas utilizando campos receptivos nos formatos 3x3 e 1x1, maxpooling com kernels 2x2 e também utilizando ReLU como função de ativação. Sua rede densa é formada por três camadas, onde as duas primeiras possuem 4096 canais e a terceira possui 1000 canais.

GoogleNet [5]: Esta rede é formada pela arquitetura de módulos Naive Inceptions, configurados a partir de grupos de convolução com kernels em 1x1(a), 3x3(b), 5x5(c) e 27 módulos de maxpooling com kernel em 3x3. No entanto, há camadas adicionais de convolução em 1x1 antes das camadas (a) e (b), porque, de acordo com o autor, essa redução de dimensão serve para melhorar a velocidade de processamento da rede.

ResNet [17]: Esta rede foi inspirada na VGG-19 e é formada por 34 camadas de convolução com kernels de 7x7 e 3x3. Este modelo introduz o conceito de blocos residuais que utilizam conexões de salto (skip connections) como proposta para resolver o problema de *vanishing gradient* [22] em redes profundas. Essa estratégia consiste em passar uma cópia da entrada para a saída como forma de melhorar a capacidade do gradiente em múltiplas camadas.

DenseNet [8]: Seguindo a linha das redes profundas, a DenseNet surgiu como outra proposta para resolver o problema de *vanishing gradient.* Este modelo é caracterizado pela inclusão de blocos densos onde a entrada de cada camada é propagada para as camadas subsequentes. Cada bloco denso é formado por camadas com kernels de 1x1 e 3x3.

SqueezeNet [7]: Esta arquitetura é formada por sucessivos módulos chamados “fire”. Esses módulos possuem dois blocos de convolução, sendo o primeiro formado por uma sequência de três camadas de convolução em 1x1. Em seguida, é alimentado para o segundo bloco de expansão formado por camadas em 1x1 e 3x3, formando assim o padrão "squeeze-and-excite" inovação deste modelo. A SqueezeNet também utiliza a função de ativação ReLU, Dropout de 50% em sua arquitetura e sua rede densa é inspirada na arquitetura NiN.

MobileNetV2 [16]: Esta rede possui uma arquitetura baseada no uso de blocos de convolução chamados de blocos residuais invertidos. Esses blocos são formados por três camadas de convolução em 1x1 (ponto a ponto), 3x3 (depthwise) e 1x1 (ponto a ponto). Ao longo de sua arquitetura, são utilizados blocos com variações de stride em 1 e 2 nas camadas depthwise, e a função de ativação ReLU6 é utilizada na primeira camada 1x1 e nas camadas 3x3.

ShuffleNetV2 [19]: Esta rede possui uma arquitetura de blocos semelhante ao padrão de bottleneck invertido. A ShuffleNetV2 também utiliza o algoritmo Channel Shuffle, herdado de sua primeira versão. Esse algoritmo é responsável por separar, distribuir e unificar uma fração de informações de cada canal do mapa de características para cada um dos canais como uma interseção de informações.

EfficientNetV1 [17]: Esse modelo melhora seu desempenho calculando a escalabilidade de três características: largura, profundidade ou resolução. E utiliza camadas de convolução semelhantes a blocos residuais invertidos na maioria das partes.

MobileNetV3 [3]: Sucessor da versão 2, a MobileNetV3 utiliza blocos de Bottleneck Invertido com variações de kernel entre 3x3 e 5x5, método "squeeze-and-excite" e h-swish como função de ativação. Em sua arquitetura, o V3 também utiliza a Pesquisa de Arquitetura Neural (NAS) como método para melhorar a precisão por meio de aprendizado por reforço.

EfficientNet V2 [18]: Este modelo incluiu um novo bloco de convolução chamado Fused-MBConv. Ao contrário de seu antecessor, este novo bloco mescla as duas primeiras camadas em apenas uma com um kernel de 3x3 e uma segunda camada com um kernel de 1x1, além de utilizar concatenação residual. Essa versão também possui a característica de aprendizado progressivo e Pesquisa de Arquitetura Neural (NAS).

B. Trabalhos relacionados

No âmbito do aprendizado profundo, há diversos algoritmos projetados para dispositivos móveis que se destacam. Esses modelos geralmente contêm poucos parâmetros treináveis e podem ser considerados como modelos rápidos e eficientes. A seguir, nós apresentamos alguns modelos que se destacam nesse contexto.

Em Xuan et al. [21] os autores propõem um modelo de CNN leve, chamado de CBCNN (Compress Binarized Convolutional Neural Network). O modelo proposto envolve a compressão tanto dos conjuntos de dados quanto das estruturas das redes neurais binarizadas para lidar com o problema de classificação multiclasse.

Eles utilizam pesos e ativações binárias no modelo e mostram experimentalmente que é possível obter uma acurácia de aproximadamente 92,94% no conjunto de dados GTSRB com apenas 0,59 milhão de parâmetros treináveis. Durante o processo de análise das imagens, o modelo CBCNN substitui a maioria das operações aritméticas por operações bitwise, reduzindo o tamanho da memória e o custo em até 32 vezes. A remoção das informações de cor também reduz significativamente o tempo de treinamento do modelo. Essas otimizações permitem que o CBCNN seja executado de forma eficaz e em tempo real em CPUs simples, em vez de GPUs.

A principal contribuição do trabalho é a introdução dos princípios de implementação de três elementos fundamentais: a função de ativação binária (Binary_act), a camada convolucional binária (Binary_C) e a camada densa binária (Binary_D). Em seguida, são apresentadas as regras de cálculo do gradiente de uma única camada no modelo CBCNN. O algoritmo descreve como calcular o gradiente de entrada e saída considerando os pesos e o gradiente atuais. Uma das limitações do trabalho é que o CBCNN apresenta uma precisão ligeiramente menor do que as redes neurais convolucionais clássicas em problemas de classificação multiclasse.

Em Huajie et al. [14] os autores propõem um modelo de CNN leve chamado DPSBC-Net (Densely Piled Steel Bar Counting-Net) e foi aplicado pelos autores na contagem de barras de aço densamente empilhadas em canteiros de obras. Esse modelo busca resolver o problema de contagem manual, que, segundo os autores, é demorado e propenso a erros, oferecendo uma solução automática usando aprendizado profundo.

A metodologia utilizada pelos autores envolve a criação da DPSBC-Net, que inclui um bloco CBAMDenseCSP e uma nova metodologia de medição da escala relativa de objetos para reduzir o número de parâmetros e o custo computacional do modelo proposto. Eles desenvolveram também um método de medição de densidade de objetos e uma "cabeça de detecção" para equilibrar as amostras positivas e negativas, melhorando significativamente o desempenho do modelo.

Os autores apresentam como inovação a melhoria do modelo mobile YoloV5. Eles introduzem um método de medição de escala relativa de objetos para eliminar caminhos redundantes de propagação de características no "neck" (parte do modelo responsável por fundir mapas de características de diferentes resoluções). Além disso, eles propõem um método de medição de densidade de objetos para eliminar o mapa de características de detecção que tem um impacto negativo no modelo proposto.

O trabalho possui como limitação o fato de que, embora o modelo seja mais rápido e preciso do que os métodos existentes, não foi mencionado se ele foi implantado em dispositivos portáteis para uso em canteiros de obras. Essa implantação prática em um dispositivo portátil é um próximo passo importante para validar a viabilidade e a eficácia do modelo em ambientes reais de construção.

Em Hao et al. [24] os autores propõem um modelo de CNN leve chamado U2ESPNet aplicado à segmentação de ramos de macieiras em pomares. Esse modelo visa melhorar a inteligência dos robôs de colheita de maçãs e possibilitar o reconhecimento de ramos com alta precisão e em tempo real, mesmo em dispositivos com recursos limitados.

A metodologia utilizada pelos autores envolveu a combinação das vantagens do Efficient Spatial Pyramid (ESP) e do U2Net para projetar a rede U2ESPNet. Essa rede possui uma estrutura de duas camadas em formato de "U", em que ambas as camadas contêm módulos ESP. Essa abordagem permite capturar informações contextuais de diferentes escalas e aumentar a profundidade arquitetônica sem aumentar excessivamente o custo computacional.

Nesse trabalho os autores propuseram um novo bloco de estrutura em forma de "U" baseado no Efficient Spatial Pyramid (ESP), chamado EUB (Encoder-stage ESP U-like Block) e DEUB (Decoder-stage ESP U-like Block), para capturar características em várias escalas da imagem de forma hierárquica através da aplicação de variações de canais e tamanhos das características de saída processadas por cada bloco.

Além disso, os autores propuseram um módulo chamado PSPP (Point-wise Simple Pyramid Pool) para a fusão de características. Esse módulo foi comparado com outro módulo de fusão de características, o PPM (Pyramid Pooling Module), utilizado no U2ESPNet. O objetivo dessas abordagens é melhorar o desempenho do modelo, permitindo uma fusão eficiente de características em diferentes escalas e reduzindo o custo computacional, memória e energia.

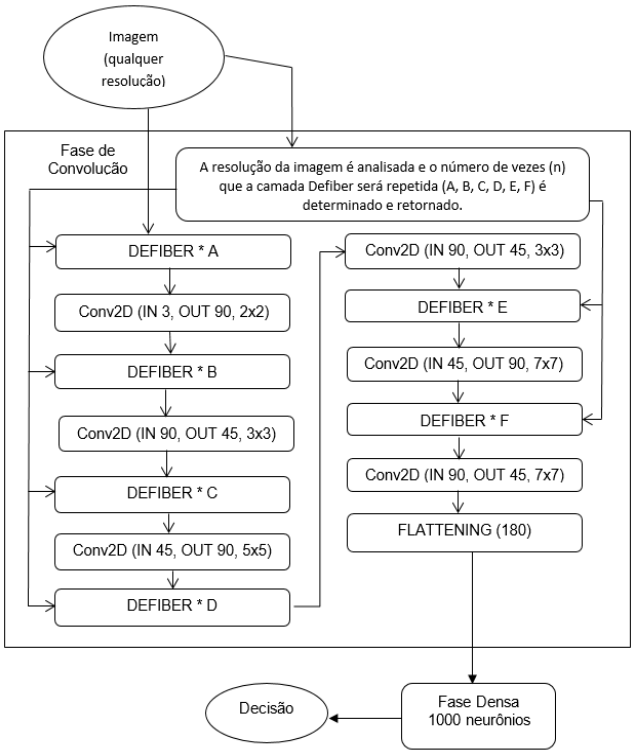
Segundo os autores, o trabalho possui a limitação de não prever ramos ocultos e não realizar a reconstrução 3D da estrutura completa em comparação com outros estudos relacionados. Essa reconstrução vai exigir uma capacidade maior do modelo o que por consequência pode dificultar sua implementação em robôs de colheita que possuam recursos limitados de hardware.

III. O MODELO PROPOSTO

A. A camada Defiber

A principal contribuição do modelo proposto é a inclusão de uma nova camada, chamada Defiber (tabela II), que está incorporada ao modelo CNN proposto, conforme apresentado na Figura 1. Seu principal objetivo é reduzir o tamanho da entrada antes de cada camada de convolução. Nesse sentido, as imagens de entrada se tornam menos complexas, levando a uma redução no número de parâmetros treináveis da rede sem prejudicar sua precisão. A **figura 3** apresenta a arquitetura geral do modelo proposto (FiberNet).

Fig. 3. Arquitetura da FiberNet.



Para descrever o funcionamento do FiberNet, a imagem de entrada entra na primeira camada Defiber (DEFIBER * A). Nesse caso, a camada Defiber reduz o tamanho da imagem de entrada removendo as linhas e colunas cujos índices são divisíveis por um limite (t). A **figura 4** apresenta uma ilustração do funcionamento da camada Defiber.

O limite (t) atualmente utilizado é três, pois foi o valor através do qual nosso modelo obteve a melhor acurácia

(acc) com a base de imagens CIFAR10. Após nossa análise empírica, foi observado que o limite três nos permitiu alcançar uma acurácia significativa, para o patamar do experimento, e ainda assim mantermos uma quantidade de repetições total mínimo (01) e com a mesma quantidade de parâmetros treináveis (tabela 1).

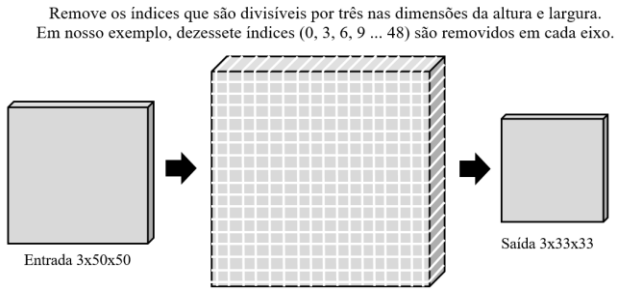
TABLE I. RESULTADO DA ANÁLISE EMPÍRICA COM VARIAÇÃO DE LIMITES PARA A CAMADA DEFIBER.

Defiber limite	Acurácia Cifar10	Repetições	Parameters
01	---	00	Error
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

TABLE II. ALGORITMO DA CAMADA DEFIBER

01:	**procedimento** Defiber
02:	**ENTRADA tensor** X
03:	**ENTRADA (inteiro)** variável Repetir-N-vezes
04:	**NOVA lista (inteiro)** NovaLista
05:	**PARA** variável **i** igual a zero até o tamanho de Repetir-N-vezes **FAÇA:**
06:	**PARA** variável **j** igual a zero até o tamanho da última dimensão do tensor X **FAÇA:**
07:	**SE** a variável **j** não é divisível Pelo limite três **(t) ENTÃO:** **ADICIONAR** variável **j** em NovaLista
08:	**FIM SE**
09:	**FIM FOR**
10:	//Retorna o tensor X redimensionado nas dimensões H e W. **RETORNA** X[:,:,NovaLista,:][:,:,:,NovaLista]
11:	**FIM procedimento**

Fig. 4. Ilustração do funcionamento da camada Defiber.



A redução realizada pela camada Defiber é repetida (n) vezes. Em seguida, a imagem reduzida é enviada para a camada de convolução. O resultado da camada de convolução é então enviado para a próxima camada Defiber (DEFIBER * B). Esse processo é realizado em todas as 12 camadas do modelo.

Além disso, como pode ser observado na f**igura 3**, as camadas Defiber possuem um parâmetro, que é o número de vezes que elas devem ser repetidas (n). Esse parâmetro é representado pelas variáveis A, B, C, D, E e F, sendo multiplicadas, por cada uma das seis camadas Defiber, respectivamente.

No início do processo de convolução, essas variáveis são definidas automaticamente após uma análise do tamanho da imagem pela função Calc-Repetições (CR), conforme descrito na tabela III e ilustrado na **figura 5**. Essa função consiste em seis estruturas de repetição (loops) aninhados onde, dentro do último loop, ocorrem simulações das camadas Defiber e de convolução do forward principal do modelo. Nessa simulação, existem cópias das 12 camadas: 06 camadas Defiber e 06 camadas de convolução com funcionalidade similar às camadas originais.

Nesta simulação é realizado o cálculo necessário para que o modelo possa prever, a partir do tamanho da imagem, quantas vezes as camadas Defiber precisarão ser repetidas ao longo do processo de convolução. É com base nesse princípio que nosso modelo consegue convolucionar imagens de qualquer tamanho.

A principal diferença entre o fluxo do forward verdadeiro e o fluxo do forward da simulação está no tipo de retorno de cada camada. No fluxo do forward verdadeiro, nós passamos um tensor que é percorrido entre as camadas, enquanto que no fluxo do forward da simulação, nós passamos uma variável do tipo inteiro (int) que representa o tamanho de uma das dimensões do tensor de entrada. Essa variável percorre todo o fluxo do forward simulado passando pelas camadas Defiber simuladas (DefiberSim) e pelas camadas de convolução simuladas (Conv2DSim).

No fluxo do forward das camadas simuladas, cada camada DefiberSim retorna uma variável do tipo inteiro que representa o tamanho final da lista (NovaLista) onde são armazenados os índices dos eixos não divisíveis pelo limite (t), como descrito na tabela IV. E por sua vez, a camada Conv2DSim recebe uma variável do tipo inteiro (recebida da camada DefiberSim anterior) e o valor dessa variável é processado através da seguinte fórmula:

output_size = (input_size + (padding_size * 2) - kernel_size) // stride_size + 1

Dessa maneira, observamos que o retorno da camada Conv2DSim, na verdade, é uma representação do processamento de um tensor pela camada de convolução (Conv2D). E com essa simulação nós conseguimos estimar, de maneira precisa, qual será o tamanho do tensor após seu processamento pela camada de convolução verdadeira. É importante salientar que a fórmula é abastecida com as mesmas informações de hiperparâmetros da Conv2D verdadeira.

Retomando a explicação sobre os loops, cada loop possui uma variável (A, B, C, D, E, F) que itera sobre uma lista contendo valores de 0 a 4. Na parte interna do último loop, essas variáveis são passadas como parâmetros para as simulações das camadas Defiber (tabela IV). A primeira camada DefiberSim recebe A, a segunda camada DefiberSim recebe B e assim por diante, até a última camada DefiberSim receber F, conforme descrito na tabela III. O fluxo continua com as camadas Defiber sendo alternadas com as Conv2DSim conforme ilustrado na

figura 5. Após a última Conv2DSim, se o valor obtido for igual a 2, a função retorna os valores das variáveis A, B, C, D, E e F para o fluxo original.

TABLE III. ALGORITMO DA FUNÇÃO CR

01:	**Procedimento** Calc-Repetições (CR)
02:	**ENTRADA tensor IN**
03:	**NOVA lista (inteiro)** NovaLista preenchida com valores de 0 a 4
04:	**NOVA (inteiro)** variável **X** preenchida com o valor da última dimensão do tensor **IN**
05:	**PARA** variável **A** igual a zero até o tamanho da lista NovaLista **FAÇA:**
06:	**PARA** variável **B** igual a zero até o tamanho da lista NovaLista **FAÇA:** **(…)** criar mais 4 loops para **C, D, E** e **F.**
07:	// Dentro do loop **F** nós começamos a simulação do forward pass. **X** e **A** são passadas como parâmetro para a primeira camada DefiberSim que retorna **X1** ao final.
08:	**X1** é passado como parâmetro pela primeira Conv2DSim, o tamanho da saída é calculado e **X2** é retornado. **X2** e **B** são passados como parâmetro para a próxima camada DefiberSim que retorna **X3.** **(…)** Repetimos esse procedimento ao longo de todas as camadas DefiberSim e Conv2DSim até que ao final **X(…)** seja retornado.
09:	**SE X(…)** for igual a 2 **ENTÃO:**
10:	**RETORNE** os valores de A, B, C, D, E e F para serem configurados nas camadas Defiber do fluxo principal.
11:	**FIM SE**
12:	**FIM PARA**
13:	**FIM do procedimento**

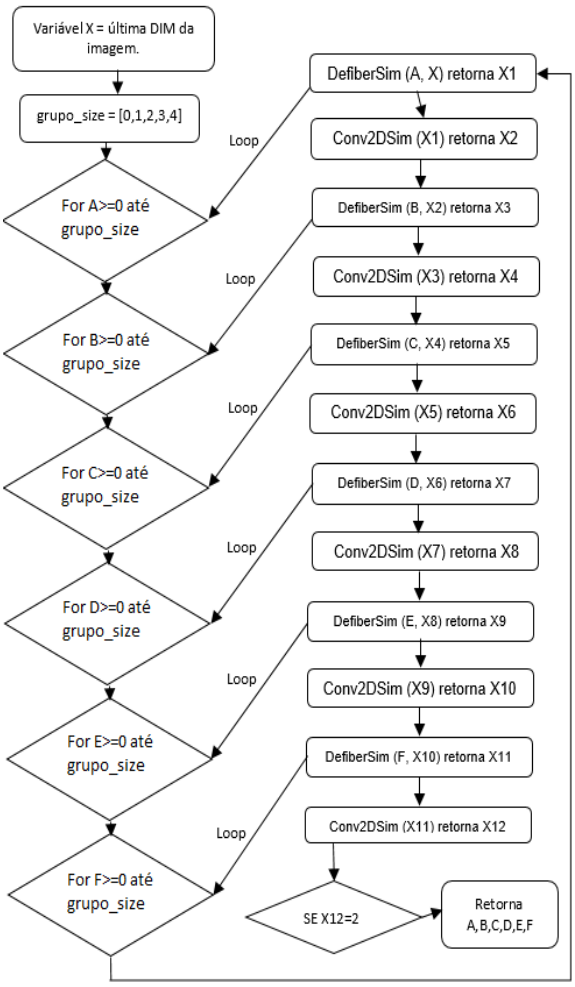
TABLE IV. ALGORITMO DA SIMULAÇÃO DA CAMADA DEFIBER.

01:	**procedimento** DefiberSim
02:	**ENTRADA variável (inteiro)** ultima-dim
03:	**ENTRADA (inteiro)** variável Repetir-N-vezes
04:	**NOVA Lista (inteiro)** NovaLista
05:	**PARA** variável **i** igual a zero até o tamanho de Repetir-N-Vezes **FAÇA:**
06:	**PARA** variável **j** igual a zero até o tamanho de ultima-dim **FAÇA:**
07:	**SE** variável **j** não é divisível pelo limite 3 (t) **ENTÃO:** **ADICIONE** variável **j** em NovaLista
08:	**END IF**
09:	**END FOR**
10:	**RETURN** size of X
11:	**END procedure**

É importante observarmos que nos loops do fluxo simulado o teste de combinação entre os valores das variáveis A, B, C, D, E, F só termina se a variável resultante tiver o tamanho 2, isto é, os loops continuarão a testar combinações entre essas seis variáveis, modificando seus valores de 0 a 4, continuamente, até que o resultado do fluxo simulado seja igual a 2. Independentemente de quais valores estejam definidos para as variáveis A, B, C, D, E e F. A natureza dessa ação baseia-se no processamento de números inteiros, o que torna o algoritmo altamente eficiente, quase sem nenhum custo computacional significativo.

Além disso, nossa análise empírica revelou que essa metodologia permite ao nosso modelo combinar valores para as variáveis A, B, C, D, E e F, mesmo em imagens com resolução de 8000x8000. Ou seja, contanto que haja hardware suficiente para processar esse tamanho de imagem, a função CR do nosso modelo será capaz de encontrar um conjunto de valores adequados para permitir que as camadas Defibers reduzam o tamanho da imagem sem prejudicar a quantidade de parâmetros treináveis da rede.

Fig. 5. Diagrama de fluxo do forward da função CR



B. *Parâmetros da camada de convolução*

Descrevendo os detalhes das camadas de convolução (Conv2D), temos um número mínimo de canais de entrada e saída configurados para manter a precisão do modelo. São os valores definidos como input (IN) e output (OUT) channels que aparecem na figura 1. Temos uma sequência de valores de kernels (2x2, 3x3, 5x5, ... 7x7) configurados com base nos valores de acurácia e quantidade de parâmetros treináveis obtidos em nossa análise empírica. Nossas análises foram realizadas com os bancos de imagens (datasets) descritos na seção 4A.

As configurações de padding, definido como zero, e stride, definido como um, seguem como padrão em todas as camadas. Além disso, nós utilizamos a normalização em lote (batchnorm) para melhorar a velocidade de aprendizado e evitar o overfitting. Por fim, aplicamos a função de ativação ReLU em todas as camadas de convolução e na única camada linear da parte densa da rede que é formada por uma camada com 1000 neurônios.

Link do código da FiberNet para consulta:
https://colab.research.google.com/drive/15-IjOcJbJudxx_RofJrVqtHHtyHbps35?usp=sharing

IV. ANÁLISE DOS EXPERIMENTOS

Para realizar essa análise empírica, é utilizado o ambiente de desenvolvimento online Google Colab (https://colab.research.google.com). Essa plataforma online fornecida pelo Google oferece um ambiente de programação compatível com Python (utilizado em deep learning) e também uma cota de espaço em memória e processamento de GPU. Além disso, nós utilizamos o framework PyTorch (https://pytorch.org/) para construir a arquitetura da CNN e toda a estrutura necessária para treinamento e teste dos modelos avaliados.

A. *Datasets utilizados*

O conjunto de dados de imagens da planta Sisal usado nesta análise empírica foi construído pelos autores deste artigo (conjunto de dados interno). As imagens de Sisal foram fotografadas com a câmera de um smartphone e possuem arquivos divididos em duas classes: fibra (subproduto da planta) e planta (imagem original de Sisal). As imagens estão configuradas com o tamanho de 200x200 pixels (**figura 6**).

O conjunto de dados de Sisal é composto por 826 imagens colorias divididas em duas classes [planta] e [fibra], com 413 imagens para cada classe. Todas as imagens de cada classe foram distribuídas aleatoriamente nas três pastas utilizadas: treinamento, validação e teste. A divisão nessas três pastas obedece à proporção 70/15/15(%) para treinamento, validação e teste, respectivamente. Essa divisão é feita de forma estratificada, na qual cada pasta contém 50% das imagens de planta e 50% das imagens de fibra. Esse procedimento é repetido, resultando em 10 datasets diferentes (dataset 01, dataset 02, dataset 03, ..., dataset 10). Os resultados apresentados na próxima seção representam os valores médios dessas 10 divisões.

O segundo conjunto de dados de imagens é a CIFAR10. Este conjunto de dados consiste em 60000 imagens coloridas de 32x32 pixels em 10 classes, com 6000 imagens por classe. É composto por várias imagens diferentes, como avião, navios, carros, gatos, cachorros, entre outros. Decidimos incluir testes com um conjunto de dados diferente, pois ele apresenta um desafio maior para os modelos analisados devido à complexidade associada ao número variado de classes e resoluções de imagem.

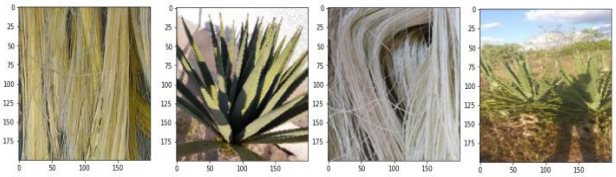
Imagens com diferentes resoluções nos permitem observar o comportamento dos modelos em cenários com mais informações (200x200) e menos informações (32x32). Além disso, a CIFAR10 é um conjunto de dados clássico e conhecido por vários pesquisadores da área. É importante salientar que a mesma metodologia utilizada com o conjunto de dados de imagens do Sisal também foi aplicada ao conjunto de dados da CIFAR10.

Os datasets, tanto Sisal quanto CIFAR10, são balanceados, contendo a mesma quantidade de imagens por classe. Para carregar as imagens, utilizamos a função DataLoader do framework Pytorch, configurando a função de embaralhamento (shuffle) como TRUE para o conjunto de treinamento e FALSE para os conjuntos de validação e teste.

Utilizamos esse recurso do embaralhamento dos dados para garantir que os modelos avaliados recebessem grupos de imagens (batchs) aleatórios do dataset. O uso dessa técnica evita que o modelo memorize a ordem específica dos dados forçando-a a aprender padrões mais gerais, o que melhora sua capacidade de generalização e torna o modelo mais robusto em relação a diferentes conjuntos de dados e situações.

A utilização de um framework com especificações estritas para o carregamento das imagens associada à igual distribuição das imagens por classe garantiu que os resultados alcançados em nossos experimentos refletem a melhor capacidade de generalização por parte dos modelos dentro do contexto de imagens estabelecidos.

Fig. 6. Exemplo da fibra e da planta do Sisal.



B. *Materiais e métodos*

O método proposto e outras nove arquiteturas de CNN (GoogleNet [5], ResNet18 [17], DenseNet121 [9], SqueezeNet [7], ShuffleNetV2 [19], MobileNetV2 [16], EfficientNetV1 [17], MobileNetV3 [3] e EfficientNetV2 [18]) serão avaliados nos conjuntos de dados descritos na Seção 4.1. Essas arquiteturas foram escolhidas por representarem o estado-da-arte das CNNs atualmente. Os critérios de avaliação são baseados nos desafios propostos por David e Paul Teich [6] para serviços baseados em IA, que incluem programabilidade, latência, acurácia, tamanho do modelo e taxa de transferência. Neste artigo, utilizaremos alguns desses critérios para avaliar a qualidade dos modelos analisados.

A latência mede o tempo de resposta de uma CNN, enquanto a taxa de transferência é o número máximo de

instâncias que um modelo pode processar em um determinado tempo. Para avaliar esses critérios, utilizamos o método descrito por Geifman [2]. Avaliamos a acurácia usando a matriz de confusão [12] para determinar as taxas de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos, e calculamos a acurácia usando a fórmula ((TP+TN)/(TP+FP+TN+FN)). Para representar a arquitetura completa dos modelos avaliados, avaliamos o número total de parâmetros treináveis (tamanho do modelo).

Também realizamos análises estatísticas utilizando os testes post-hoc de Friedman [15] e Nemenyi [20]. Formulamos a hipótese nula h0 de que todos os grupos de amostras são equivalentes e a hipótese alternativa h1 de que um ou mais grupos de amostras vêm de populações diferentes. A rejeição da hipótese nula com valor $p \leqslant \alpha$ (0,05) indica que a hipótese alternativa é verdadeira. O teste de Nemenyi foi então aplicado para análise em pares, sendo considerado como similar um valor de $p \geqslant 0,05$.

Para realizar a fase de treinamento dos modelos, nós utilizamos um notebook online disponível no Google Colab. No momento da realização dos experimentos nós utilizamos a seguinte configuração: processador T4 com 10 GB de GPU e 3 GB de memória RAM.

C. Hiperparâmetros de treinamento

Hiperparâmetros são parâmetros configuráveis externamente que controlam o comportamento de um algoritmo de aprendizado de máquina. Ao contrário dos parâmetros do modelo, que são aprendidos durante o treinamento, os hiperparâmetros são definidos pelo desenvolvedor antes do processo de treinamento e afetam diretamente o desempenho e a capacidade de generalização do modelo.

Em nossa pesquisa nós utilizamos um conjunto de hiperparâmetros que foram definidos, antes das sessões de treinamento, levando em consideração as restrições do hardware e os ajustes realizados durante o processo de análise empírica. Nesse processo nós ajustamos as configurações gradualmente até encontrarmos os melhores valores para cada um dos hiperparâmetros ajustados. Nos próximos parágrafos, nós descrevemos os detalhes sobre os hiperparâmetros utilizados, valores definidos para nossa pesquisa e o impacto esperado nos modelos avaliados.

O tamanho do lote (batch-size) é um hiperparâmetro que controla o número de amostras de treinamento processadas em uma única atualização dos pesos do modelo durante o treinamento. Ele define o tamanho dos grupos de dados, chamados lotes, que são utilizados para calcular os gradientes e ajustar os pesos da rede. Em nossa pesquisa nós utilizamos um valor de batch-size para a base do Sisal e um valor para a base CIFAR10.

Para a base do Sisal nós observamos que, por conta de restrições de hardware, quando nós utilizávamos valores muito grandes o sistema apresentava falhas. Dessa forma restou definida o valor 04 para o treinamento com essa base. Já com a base CIFAR10 nós começamos a análise a partir do valor 08 e seguimos aumentado o valor, com múltiplos de 08, até o valor limite que foi de 64.

Com os valores utilizados até o limite nós não observamos nenhum decréscimo de acurácia e o tempo de treinamento, com cada modelo, continuou relativamente igual. Os resultados apresentados na seção V refletem o processo de inferência dos modelos treinados com este valor.

Adicional a este hiperparâmetro, nós temos também um segundo batch-size que é utilizado pelo método que avalia a taxa de transferência dos modelos, cuja descrição está na seção IVb. Para esse batch-size nós também utilizamos o valor limite de 64 e os resultados obtidos para este critério, descritos na seção V, expressam a quantidade de dados que podem ser transferidos pelos modelos em um determinado período de tempo.

As épocas são uma unidade de medida que representa uma passagem completa por todo o conjunto de dados de treinamento durante o processo de aprendizado do modelo. Em cada época, o algoritmo de treinamento utiliza todo o conjunto de dados para atualizar os pesos das conexões entre os neurônios da rede, buscando minimizar o erro entre as previsões do modelo e os rótulos corretos dos dados de treinamento.

Uma vez que não há uma regra estabelecida na literatura para determinar a quantidade ideal de épocas para o treinamento e também considerando a limitação de hardware disponível, optamos por fixar o limite de 50 épocas para treinar todos os modelos em nossa pesquisa. Essa escolha foi fundamentada na constatação de que, durante a análise empírica, todos os modelos apresentaram um desempenho notavelmente estável com este valor. Assim, buscamos equilibrar a eficiência do treinamento com a garantia de uma boa capacidade de generalização dos modelos, considerando as limitações do ambiente de processamento.

Além disso, optamos por conduzir o experimento de forma imparcial, assegurando que nenhum modelo fosse favorecido em detrimento dos outros, uma vez que todos foram treinados com exatamente as mesmas configurações de hiperparâmetros. Essa abordagem nos proporcionou resultados comparáveis e confiáveis, permitindo uma avaliação mais justa e precisa do modelo proposto e concorrentes.

Outro hiperparâmetro utilizado foi à taxa de aprendizado. Ela representa o tamanho do passo que a rede neural dá em direção ao mínimo local durante o processo de otimização dos pesos. Em outras palavras, a taxa de aprendizado controla o quão rápido o modelo ajusta seus parâmetros em resposta ao erro calculado durante o treinamento.

Adicionalmente à taxa de aprendizado nós utilizamos também um otimizador adaptativo que é um tipo de algoritmo usado para ajustar os parâmetros da rede durante o processo de treinamento. Esse algoritmo ajusta a taxa de aprendizado individualmente para cada parâmetro, permitindo uma adaptação mais precisa e dinâmica durante o treinamento. Esse otimizador leva em consideração o histórico dos gradientes calculados durante o treinamento para determinar a taxa de aprendizado para cada parâmetro.

Dessa forma, parâmetros que estão convergindo mais lentamente ou com oscilações podem receber uma taxa de aprendizado menor, enquanto parâmetros que estão progredindo rapidamente podem receber uma taxa de

aprendizado maior. Em nossa pesquisa nós utilizamos o otimizador Adam (Adaptive Moment Estimation) através do qual nós escalonamos o valor da taxa de aprendizado.

Em nossa pesquisa nós começamos a análise empírica com a variação da taxa de aprendizado a partir do valor de 0,1 até o valor definido, e com o qual treinaríamos os modelos, de 0,0001. Com esse valor nós observamos que houve uma boa convergência dos modelos atingindo o ápice das acurácias sem desestabilizar a curva de aprendizado do gradiente.

Por fim, dentre os hiperparâmetros mais relevantes, nós utilizamos também uma função de custo (criterion). Essa função é responsável por calcular a diferença entre as previsões feitas pelo modelo e os rótulos verdadeiros dos dados de treinamento. O objetivo do treinamento é minimizar essa função de perda para que o modelo seja capaz de fazer previsões mais precisas nos dados de teste.

Uma vez que nossa pesquisa se trata de um problema de classificação nós escolhemos a função de perda de entropia cruzada (Cross-Entropy) que mede a discrepância entre as probabilidades de classe previstas e as probabilidades de classe verdadeiras.

Em nossa pesquisa, buscamos avaliar os modelos de forma cuidadosa e eficiente. Portanto, duas importantes decisões foram tomadas: optamos por não utilizar nenhum tipo de aumento de dados (data augmentation) e todos os modelos foram treinados sem utilizar inicialização de pesos. Essas escolhas foram fundamentadas em suas vantagens específicas. Ao não aplicar o aumento de dados, permitimos que os modelos fossem treinados exclusivamente com os dados originais, aproveitando a representatividade do conjunto de dados em sua forma natural.

Essa abordagem foi especialmente útil quando o conjunto de dados original continha informações suficientes para a tarefa em questão. Além disso, ao evitar a inicialização de pesos, garantimos que os modelos começaram o treinamento com valores mais neutros, o que reduziu possíveis vieses ou estados desfavoráveis.

Dessa forma, o aprendizado ocorreu de maneira mais imparcial, permitindo a descoberta eficiente de padrões relevantes nos dados. Vale ressaltar que essas decisões foram consideradas em função da natureza específica da tarefa e do conjunto de dados utilizados em nossa pesquisa, visando maximizar a eficácia na avaliação dos modelos (desenvolvido e concorrentes).

V. RESULTADOS DOS EXPERIMENTOS

A. Base do Sisal

Os resultados de acurácia de todos os 10 métodos analisados estão apresentados na **Tabela 5**. Além disso, os resultados de taxa de transferência (TT) e tempo de inferência (TI) estão apresentados na **Tabela 6**, enquanto os resultados de parâmetros treináveis estão apresentados na **Tabela 7**.

Utilizando a acurácia como base, os resultados dos outros três critérios são apresentados: parâmetros treináveis como linha vermelha, TT como linha amarela e TI como linha verde. Para criar um ambiente de comparação, os valores desses três últimos critérios foram convertidos em uma escala percentual, a partir dos valores mais altos de cada critério.

TABLE V. RESULTADO DA ACURÁCIA PARA A BASE DO SISAL.

	Models	**Accuracy**
01	SqueezeNet	98.96%
02	DenseNet	98.65%
03	ResNet	98.02%
04	FiberNet	96.25%
05	GoogleNet	95.94%
06	EfficientNetV2	94.79%
07	MobileNetV2	83.23%
08	ShuffleNet	81.25%
09	MobileNetV3	73.75%
10	EfficientNetB0 (V1)	69.48%

TABLE VI. RESULTADO DE TT E TI PARA A BASE DO SISAL.

	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 VII. MODELOS E SEUS RESPECTIVOS PARÂMETROS E TAMANHO DE ARQUIVO.

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

De forma geral, ao analisar uma arquitetura de CNN, é necessário encontrar um equilíbrio entre desempenho (acurácia) e eficiência (TI, TT e parâmetros treináveis). Por exemplo, ao comparar DenseNet e FiberNet, em termos de acurácia, DenseNet obteve uma precisão 2,4 pontos percentuais (p.p.) maior do que a FiberNet. No entanto, FiberNet é muito mais eficiente do que DenseNet, sendo 96,35 p.p. mais rápido em termos de TI e tendo 96,95 p.p. mais instâncias (TT).

Por fim, a FiberNet possui 91,16 p.p. menos parâmetros do que DenseNet. Em outras palavras, FiberNet alcançou um desempenho ligeiramente inferior ao de DenseNet, mas obteve resultados muito melhores em termos de eficiência (TT, TI e parâmetros treináveis).

A diferença proporcional entre esses resultados pode ser explicada pela diferença de paradigmas adotados entre esses dois modelos. Na DenseNet os autores buscaram por uma resposta ao desafio da classificação de imagens a partir da inclusão de muitas camadas sucessivas. Esse tipo de arquitetura aprofunda a busca por informações relevantes na imagem redimensionando-a mediante a passagem por diversas telas de convolução.

Esse processo ainda conta com uma quantidade massiva de canais aplicados por camada na tentativa de realçar algumas características relevantes. Dessa maneira, nós podemos observar que modelos que adotam essas características: (a) profundidade excessiva nas camadas de convolução, (b) grande número de canais de entrada nos filtros de convolução e a (c) adoção de camadas residuais são modelos que aderem a um paradigma que vincula sua eficiência à profundidade da rede. Independentemente se ela é ou não considerada como mobile, por exemplo, como a SqueezeNet, a MobileNet ou a ShuffleNet.

A adoção desse tipo de paradigma, alicerçado na busca por aprendizado em profundidade é de certa forma muito mais custosa do que, por exemplo, no paradigma que adotamos para a FiberNet. Nosso modelo foca na redução gradual da imagem (input) antes de cada camada de convolução, através de uma nova camada denominada Defiber, conseguindo com isso que a carga de processamento por camada seja significantemente menor. Seria, portanto, um paradigma da redução gradual alicerçada na redução do trabalho para as camadas de convolução.

Ao compararmos as arquiteturas de CNN com os melhores desempenhos, pode-se observar que a FiberNet obteve o melhor resultado de parâmetros treináveis, seguido por SqueezeNet (+39,57 p.p.), ShuffleNet (+66,89 p.p.), MobileNetV3 (+70,33 p.p.) e MobileNetV2 (+78,47 p.p.).

E com isso, também podemos observar que o pequeno número de parâmetros treináveis entregue pela FiberNet não impactou negativamente nos resultados da acurácia. Isso ocorre porque todos os seis modelos que obtiveram resultados de acurácia próximos a 95% (SqueezeNet, DenseNet, ResNet, FiberNet, GoogleNet e EfficientNetV2) apresentaram uma diferença média de acurácia inferior a 3 p.p.

Assim sendo, podemos afirmar, portanto, que para a tarefa de classificação das imagens do Sisal, a quantidade superior de camadas de convolução e recursos, presentes nos modelos concorrentes, não resultou em uma superioridade na acurácia. Na perspectiva do confronto entre os dois paradigmas, no quesito da relação acurácia/parâmetros treináveis, restou provado que o paradigma da FiberNet foi tão bom quanto os demais.

Pois, ao contrário de outros modelos de CNN que geralmente obtêm ganhos de aprendizado a partir de camadas sucessivas em profundidade, aplicando um grande número de canais e usando kernels de diferentes tamanhos, nosso modelo foi capaz de obter aprendizado a partir da aproximação de áreas de interesse, reduzindo gradualmente a entrada.

Na mesma perspectiva, pode-se observar que houve uma redução considerável no número de instâncias (taxa de transferência - TT) processadas pelos outros modelos, em comparação com FiberNet: MobileNetV3 (-42,06 p.p.), SqueezeNet (-87,43 p.p.), MobileNetV2 (-90,41 p.p.) e ShuffleNet (-83,87 p.p.).

A mesma observação ocorre para os resultados de taxa de inferência (TI), nos quais houve uma diminuição na velocidade de processamento entregue pela FiberNet, em comparação com MobileNetV3 (+63,5 p.p.), SqueezeNet (+66,02 p.p.), MobileNetV2 (+81,64 p.p.) e ShuffleNet (+84,6 p.p.).

Nesses quesitos de TT e TI fica ainda mais fácil para observarmos os contrastes entre a aplicação dos dois paradigmas citados anteriormente. A FiberNet é um modelo extremamente simples e que possui em sua arquitetura uma estrutura básica, com convoluções comuns e pouca profundidade. A arquitetura da FiberNet sequer possui blocos de convolução, ao contrário dos modelos concorrentes que, de igual entre eles, possui em sua arquitetura algum tipo de combinação específica de conjunto de camadas que utilizam variações de kernels (blocos de convolução) na fase de convolução da rede.

Essa opção de arquitetura, associada à adoção do paradigma da profundidade, deveria tornar esses modelos concorrentes relativamente melhores do que a FiberNet, mas o que observamos é que, no caso específico da tarefa de classificação da base de imagens do Sisal, esse fato não ocorreu.

Nesse ponto em específico poder-se-ia até indagar que a relação arquitetura/velocidade teria sofrido algum tipo de interferência por conta da base de dados: qualidade das imagens, quantidade de classes, etc. Mas para contrapor essas indagações é que nos lembramos da nossa escolha metodológica por utilizarmos dois bancos de dados de imagens (datasets) com diferentes graus de dificuldade. Mais adiante, ao analisarmos os resultados dos modelos com a base de dados CIFAR10 retornaremos a este ponto.

Portanto, os resultados obtidos nas **Tabelas 5, 6 e 7** demonstraram que a aplicação da camada Defiber associada a um pequeno número de camadas de convolução não impactou negativamente os resultados de acurácia do FiberNet. Além disso, as configurações atuais dos hiperparâmetros utilizados no FiberNet foram suficientes para atingir uma acurácia equivalente aos melhores modelos de CNN.

Para apoiar os resultados obtidos, agora apresentamos os resultados dos testes estatísticos com o conjunto de dados sisal. Começamos apresentando os resultados do teste de Friedman (**Tabela 8**). Uma vez que o valor-p foi menor que 0,05, para todos os três critérios, rejeitamos a hipótese nula (h0) e aplicamos o teste post-hoc para determinar, por meio de uma comparação em pares, quais pares são mais relevantes.

O resultado post-hoc de Nemenyi para o critério de acurácia é apresentado na **Tabela 9**. Nessa tabela, comparamos apenas os resultados do valor-p do melhor resultado (SqueezeNet) em relação aos outros modelos.

TABLE VIII. RESULTADO DO TESTE DE FRIEDMAN PARA ACURÁCIA, TI E TT DA BASE SISAL.

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

TABLE IX. RESULTADO POST-HOC DA ACURÁCIA DA BASE SISAL.

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

De acordo com a análise estatística, os seguintes modelos de CNN: SqueezeNet, FiberNet, DenseNet, ResNet, GoogleNet e EfficientNetV2 podem ser considerados equivalentes, pois todos obtiveram um valor-p $\geqslant$ 0,05. Concluindo os resultados do conjunto de dados sisal, a **Tabela 10** apresenta os resultados do teste post-hoc de Nemenyi para os critérios de taxa de transferência e tempo de inferência.

TABLE X. POST-HOC DE TT E TI PARA A BASE SISAL.

	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

Com base nos critérios de taxa de inferência e taxa de transferência, o teste de Nemenyi revelou que os resultados dos modelos FiberNet, ShuffleNet, MobileNetV2, SqueezeNet e MobileNetV3 podem ser considerados equivalentes. O desempenho superior do FiberNet pode ser atribuído à sua menor quantidade de parâmetros treináveis e a redução prévia da entrada pela camada Defiber proporcionou uma vantagem quantitativa que teve um impacto positivo no funcionamento do FiberNet.

B. Base da CIFAR10

Nesta seção, serão apresentados os resultados do conjunto de dados CIFAR10. Os valores médios de acurácia são apresentados na **Tabela 11**, os resultados de TT e TI são apresentados na **Tabela 12**. O número de parâmetros treináveis para os modelos em ambos os conjuntos de dados permaneceu inalterado (**Tabela 08**).

A partir dessas tabelas e da figura, pode-se observar que nenhum modelo de CNN obteve uma acurácia igual ou superior a 95%. Os melhores resultados de acurácia foram obtidos pelo GoogleNet (75,9%), seguido pelo FiberNet (-1 p.p.), ResNet (-12,5 p.p.), DenseNet (-14,5 p.p.), SqueezeNet (-17,5 p.p.) e EfficientNetV2 (-18,7 p.p.). Um padrão de comportamento semelhante ao conjunto de dados sisal também foi obtido neste conjunto de dados.

Em termos dos critérios de eficiência, considerando que o número de parâmetros treináveis permanece o mesmo obtido na seção anterior, observamos que houve uma oscilação nos valores de TT para os dez modelos. O melhor resultado de TT foi alcançado pelo MobileNetV3, seguido pelo ShuffleNet (-25,76 p.p.), SqueezeNet (-49,41 p.p.), FiberNet (-53,09 p.p.), ResNet (-53,68 p.p.) e MobileNetV2 (-63,64 p.p.).

Também observamos uma variação nos resultados de IT, com o melhor resultado sendo entregue pelo SqueezeNet, seguido pelo FiberNet (+18,56 p.p.), ResNet (+37,86 p.p.), MobileNetV3 (+57,84 p.p.), MobileNetV2 (+62,73 p.p.) e ShuffleNet (+69,13 p.p.).

TABLE XI. RESULTADO DA ACURÁCIA PARA A BASE CIFAR10.

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 XII. RESULTADO DE TT E TI PARA A BASE CIFAR10.

	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

Portanto, considerando os resultados obtidos com este conjunto de dados, podemos concluir que o uso da camada defiber associada a um ambiente convolucional reduzido não influenciou negativamente o processo de predição do modelo, uma vez que a FiberNet apresentou um resultado promissor em termos de acurácia e eficiência comparáveis.

Ao comparar os resultados dos dois conjuntos de dados em termos de acurácia, observamos que os mesmos seis modelos de CNN que obtiveram bons níveis de acurácia no conjunto de dados do Sisal também alcançaram bons níveis de acurácia neste conjunto de dados. Em termos dos resultados de eficiência, o modelo de CNN proposto conseguiu estar entre os cinco melhores resultados em ambos os critérios.

Para considerarmos, portanto, a discussão dos resultados obtidos com a base do Sisal e com a base CIFAR10 é pertinente, retomando o ponto da discussão da subseção anterior, relembrar que está segunda base é formado por uma quantidade de imagens e de classes muito maior do que a base de imagens do Sisal. Fica claro, portanto, que o desafio de classificação nesse cenário aumentou significantemente. E ainda há o quesito resolução, pois a base CIFAR10 é formada por imagens com resolução de 32x32 pixels, ou seja, são imagens muito menores do que as imagens da base Sisal e apresentam um grau de desafio de classificação maior, pois cada imagem possui uma quantidade menor de informações (pixels) disponíveis para os modelos analisarem.

Com base nos resultados obtidos em relação à acurácia na base CIFAR10, é notável que o nosso modelo, a FiberNet, desenvolvido a partir de uma arquitetura simples, alcançou resultados tão bons quanto os modelos

comparados na subseção anterior. Além disso, é importante destacar que a FiberNet obteve esses resultados com a mesma quantidade de parâmetros treináveis em ambos os cenários. Isso ressalta a eficácia da nossa abordagem, demonstrando que é possível alcançar um desempenho significativo mesmo com uma arquitetura mais simples e sem aumentar a complexidade dos modelos, pois o nosso modelo manteve a mesma quantidade de parâmetros treináveis em ambos os testes.

Os cinco modelos que apresentaram bons resultados em relação à acurácia na base Sisal também obtiveram desempenho satisfatório na base CIFAR10. Contudo, é importante destacar que nem todos os modelos considerados como mobile tiveram um desempenho igualmente bom. Nesse contexto, surge a relevância de considerarmos o tempo de treinamento da rede.

Ao analisarmos estatisticamente o desempenho da FiberNet em comparação com os outros modelos, observamos que nosso modelo ficou no mesmo patamar. Isso nos leva a concluir que a FiberNet, nas mesmas condições de treinamento, consegue alcançar resultados significativamente equivalentes ou muito próximos aos modelos concorrentes. Esse fator reforça a eficiência da nossa abordagem, demonstrando que é possível obter um desempenho competitivo com um tempo de treinamento semelhante aos modelos concorrentes, mesmo mantendo a simplicidade da arquitetura.

É prematuro afirmar que essa condição se repetirá ao apresentarmos a FiberNet a qualquer outro conjunto de imagens, pois no desafio de classificação de imagens muitos são os fatores que contribuem para o desequilíbrio de resultados entre diferentes bases. Além disso, há fatores externos, como configuração de hardware, que também podem influenciar nos resultados.

Contudo, é importante destacar que desenvolver algoritmos de CNN a partir de conceitos simples, como os da camada Defiber e de arquiteturas igualmente simples, como a FiberNet, não significa necessariamente que teremos uma rede inferior em termos de qualidade. É possível obter um bom desempenho de classificação mesmo com uma rede significativamente menor.

Continuando, para validar os resultados obtidos neste conjunto de dados, foi realizada também uma análise estatística e a **Tabela 13** apresentam os resultados do teste de Friedman. Com base no resultado do teste de Friedman, rejeitamos a hipótese nula (h0) para todos os três critérios. Em seguida, os testes post-hoc de Nemenyi foram aplicados para acurácia (**Tabela 14**), TT e TI (**Tabela 15**).

Pelos resultados de acurácia, TT e TI, podemos observar que o modelo de CNN proposto se manteve entre os cinco melhores métodos. Além disso, ele pode ser considerado estatisticamente equivalente aos melhores modelos em todos os critérios. Isso demonstra que o método proposto conseguiu reduzir a complexidade computacional de um modelo de CNN sem prejudicar sua acurácia.

TABLE XIII. RESULTADO DE FRIEDMAN PARA ACC, TT E TI DA BASE CIFAR10.

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

TABLE XIV. RESULTADO POST-HOC PARA ACC DA BASE CIFAR10.

Ordem	Modelos	Acurácia (acc)
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 XV. RESULTADO POST-HOC PARA TT E TI DA BASE CIFAR10.

	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

VI. CONSIDERAÇÕES FINAIS

Este artigo apresenta a FiberNet, um modelo de CNN compacto e rápido com poucos parâmetros treináveis. Sua principal novidade é a implementação de uma nova camada denominada Defiber que reduz o tamanho das imagens, preservando suas informações. Embora inicialmente desenvolvido para classificar uma planta específica, a eficiência da FiberNet transcende esse propósito. O modelo processa imagens com rapidez e precisão, sem demandar recursos computacionais excessivos. A viabilidade do método proposto foi avaliada por meio de análise empírica.

Os resultados dessa avaliação demonstraram que a CNN proposta possui apenas 754.345 parâmetros treináveis e alcançou uma precisão de 96,25% na classificação da planta do Sisal. Ao compararmos com o amplamente reconhecido conjunto de dados CIFAR10, a FiberNet obteve o segundo melhor desempenho em precisão, ficando ligeiramente abaixo do GoogleNet (com apenas 1 ponto percentual de diferença). Esses resultados ressaltam a eficácia e o potencial da FiberNet em lidar com diferentes conjuntos de dados, evidenciando a competitividade do modelo em relação aos algoritmos de referência.

Importante notar que a FiberNet é consideravelmente mais eficiente que o GoogleNet em termos dos critérios de TT (tempo de treinamento), TI (tempo de inferência) e parâmetros treináveis. Essa promissora eficiência é alcançada devido ao uso da camada Defiber, que reduz o tamanho da imagem de entrada antes de cada camada de convolução. Essa abordagem destaca a vantagem da FiberNet e reforça seu potencial como uma solução viável e eficiente para aplicações de classificação de imagens em

cenários de uso com hardwares de baixo custo ou de pouco poder de processamento.

Como continuação de nossa pesquisa, pretendemos analisar o impacto de nossa metodologia em um modelo de CNN com diferentes configurações de hiperparâmetros. Além disso, pretendemos avaliar seu desempenho em conjuntos de dados de imagens do sisal com diferentes resoluções de imagem [23]. Também pretendemos analisar os resultados da FiberNet sem a camada Defiber, substituindo-a por camadas de pooling e comparando os resultados com os resultados atuais do FiberNet.

VII. REFERÊNCIAS

[1] A. AGRAWAL, and N. MITTAL. 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, 2020.

[2] A. GEIFMAN. The Correct Way to Measure Inference Time of Deep Neural Networks. May 05, 2020. Available at: "https://towardsdatascience.com/the -correct-way-to-measure-inference-time-of-deep-neural-networks-304a54e5187f" (Accessed on: March 22, 2021).

[3] A. HOWARD et al. Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019. p. 1314-1324.

[4] A. KRIZHEVSKY, I. SUTSKEVER and GE. HINTON. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, v. 25, p. 1097-1105, 2012.

[5] C. SZEGEDY, et al. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 1-9.

[6] DA. TEICH and PR. TECH. PLASTER: A Framework for Deep Learning Performance. Tech. rep. TIRIAS Research, 2018.

[7] FN. IANDOLA. et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360, 2016.

[8] GE. HINTON, N. SRIVASTAVA, A. KRIZHEVSKY, I. SUTSKEVER and RR. SALAKHUTDINOV. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

[9] G. HUANG, Z. LIU, LVD. MAATEN and KQ. WEINBERGER. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 4700-4708.

[10] H. GHOLAMALINEZHAD and H. KHOSRAVI. Pooling Methods in Deep Neural Networks, a Review. arXiv preprint arXiv:2009.07485, 2020.

[11] HOWARD, Andrew G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[12] K. PEARSON. On the theory of contingency and its relation to association and normal correlation. Drapers' Company Research Memoirs. Biometric series I: Dulau and Co, 1904.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[14] LIU, Huajie et al. Lightweight convolutional neural network for counting densely piled steel bars. Automation in Construction, v. 146, p. 104692, 2023.

[15] M. FRIEDMAN. 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, 1937.

[16] M. SANDLER, A. HOWARD, M. ZHU, A. ZHMOGINOV and L. CHEN. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. p. 4510-4520.

[17] M. TAN and Q. LE. EfficientNet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. PMLR, 2019. p. 6105-6114.

[18] M. TAN and Q. LE. Efficientnetv2: Smaller models and faster training. In: International Conference on Machine Learning. PMLR, 2021. p. 10096-10106.

[19] N. MA, X. ZHANG, H. ZHENG and J. SUN. ShuffleNet v2: Practical guidelines for efficient CNN architecture design. In: Proceedings of the European conference on computer vision (ECCV). 2018. p. 116-131.

[20] PB. NEMENYI. Distribution-free multiple comparisons. Princeton University, 1963.

[21] QI, Xuan et al. A Lightweight Binarized Convolutional Neural Network Model for Small Memory and Low-Cost Mobile Devices. Mobile Information Systems, v. 2023, 2023.

[22] S. HOCHREITER. 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, 1998.

[23] V. THAMBAWITA, et al. Impact of Image Resolution on Deep Learning Performance in Endoscopy Image Classification: An Experimental Study Using a Large Dataset of Endoscopic Images. Diagnostics, v. 11, no. 12, p. 2183, 2021.

[24] WAN, Hao et al. U2ESPNet—A lightweight and high-accuracy convolutional neural network for real-time semantic segmentation of visible branches. Computers and Electronics in Agriculture, v. 204, p. 107542, 2023.