REVISTA ENERGIA NA AGRICULTURA

DOI: http://dx.doi.org/10.17224/EnergAgric.2018v33n3p284-291

ISSN: 1808-8759 (cd-rom)
2359-6562 (on-line)

DESEMPENHO DA TÉCNICA *DEEP LEARNING* NA ANÁLISE E CATEGORIZAÇÃO DE IMAGENS DE DEFEITO DE MADEIRA

Matheus Henrique Baldi de Almeida[1], Roger Cristhian Gomes[2], Osvaldo Cesar Pinheiro de Almeida[3] & Adriano Wagner Ballarin[4]

RESUMO: O uso intensivo da inteligência artificial tem contribuído para o avanço em diversas áreas como a de análise de imagens digitais com o emprego de aprendizado de máquina. Para a análise e categorização dessas imagens destaca-se a técnica que utiliza redes neurais artificiais, envolvendo o estudo específico, a extração das características por meio da análise de dados da imagem do objeto e a especificação de qual será o impacto dessas características no modelo neural para cada uma das categorias, o que exige a imersão do pesquisador em uma área ou campo de pesquisa que não é normalmente de seu domínio. A *Deep Learning*, técnica que utiliza redes neurais convolucionais artificiais, tem a capacidade de aprender e extrair as características durante o treinamento, sem a especificação dessas características no modelo, geralmente apresentando resultados melhores do que os observados por modelos de redes neurais que tiveram as características observadas e programadas por humanos. O objetivo desse trabalho foi aplicar a *Deep Learning* com auxílio da linguagem de programação Python e duas bibliotecas (Keras e NumPy) na categorização de um conjunto de imagens de tábuas de madeira, avaliando seu desempenho. Foram elaborados e avaliados alguns modelos de rede neural convolucional aplicados nesse processo, com resultados promissores; o melhor deles apresentou erro de categorização na ordem de cinco por cento.

PALAVRAS-CHAVE: Análise de dados. Aprendizado de Máquinas. Extração de Características. Redes Convolucionais Artificiais. Keras.

DEEP LEARNING PERFORMANCE IN WOOD DEFECT IMAGES ANALYSIS AND CATEGORIZATION

ABSTRACT: The intensive use of artificial intelligence has contributed to the advancement in several areas such as the analysis of digital images with the use of machine learning. For the analysis and categorization of these images, we highlight the technique that uses artificial neural networks, involving the specific study, the extraction of the characteristics through the analysis of data of the image of the object and the specification of what will be the impact of these characteristics in the neural model for each of the categories, which requires the immersion of the researcher in an area or field of research that is not normally of his domain. Deep Learning, a technique that uses artificial convolutional neural networks, has the ability to learn and extract characteristics during training, without specifying these characteristics in the model, generally presenting better results than those observed by neural network models that had the characteristics observed and programmed by humans. The objective of this work was to apply Deep Learning using Python programming language and two libraries (Keras and NumPy) in the categorization of a set of images of wood boards, evaluating their performance. Some models of convolutional neural network applied in this process were developed and evaluated, with promising results; the best of them presented categorization error in the order of five percent.

KEYWORDS: Artificial Convolutional Neural Networks. Data Analysis. Feature Extraction. Machine Learning. Keras.

1 INTRODUÇÃO

A atividade de graduação e classificação de madeiras a partir de seus defeitos é cansativa e repetitiva e, quando executado por humanos – não automatizada – resulta normalmente em resultados pouco confiáveis. Segundo Kline, Surak e Araman (2003), a taxa de acerto de graduadores humanos é de 48% na linha de desdobramento, demonstrando esses resultados insatisfatórios. Por esta razão, métodos de classificação automatizada por imagens foram desenvolvidos visando

[1] Tecnólogo em Análise e Desenvolvimento de Sistemas, FATEC, matheushbaldi@gmail.com
[2] Mestre em Agronomia (Energia na Agricultura), Faculdade de Ciências Agronômicas, UNESP, rcristh@hotmail.com
[3] Doutor em Agronomia (Energia na Agricultura), Faculdade de Ciências Agronômicas, UNESP, cesar@fatecbt.edu.br
[4] Doutor em Engenharia Civil (Engenharia de Estruturas), USP, adriano.ballarin@unesp.br

diminuir esse problema, atingindo resultados promissores, tais como, taxas de acerto na classificação de 90,5% (RALL, 2010) e 96,9% (ALMEIDA, 2014). Novos métodos de processamento e classificação de imagens têm surgido ao longo do tempo, sendo que muitos deles utilizam técnicas de inteligência artificial.

A inteligência artificial tem sido tema cada vez mais recorrente no desenvolvimento de ferramentas que tanto auxiliam na tomada de decisões quanto são capazes de decidir de forma autônoma, com base nos dados de entrada que recebem. Estudos realizados têm proporcionado grandes avanços em várias áreas de aplicação que vão muito além da informática, graças aos padrões que as máquinas conseguem reconhecer em imagens e outras fontes.

A análise de imagens é um campo bastante explorado nesse quesito. Existem muitos *softwares* desenvolvidos especificamente para tratar imagens digitais e aplicar algoritmos que façam a extração dos dados de interesse contidos nessas imagens. Tais *softwares* instruem o computador a seguir uma sequência de ações que utilizará categorias pré-definidas na programação, havendo um estudo minucioso para a definição das características relevantes a serem exploradas e utilizadas no processamento.

O aprendizado de máquina tem como objetivo fazer com que os computadores sejam programados para aprender como processar os dados que estão sendo inseridos (SHALEV-SHWARTZ; BEN-DAVID, 2014), ou seja, em vez de os passos serem previamente programados, os computadores aprendem a extrair características e usá-las para determinado fim, tal como a análise e categorização de imagens.

Existem algumas formas de desenvolver programas capazes de aprender sozinhos, como a utilização de redes neurais artificiais, que são modelos computacionais compostos de unidades de processamento adaptáveis e com conexões densas que fornecem uma grande variedade de soluções para problemas, tais como classificação de padrões, síntese e análise de padrão de voz, compreensão de imagens, previsões e outros (HASSOUN, 1995). Essa capacidade de auto aprendizado ocorre devido ao fato de as redes neurais artificiais serem modelos matemáticos que buscam imitar o funcionamento de uma estrutura de rede neural biológica (GOODFELLOW; BENGIO; COURVILLE, 2016).

Nos últimos anos, uma abordagem relacionada com o aprendizado de máquina e redes neurais tem ganhado muita força. A *Deep learning*, ou aprendizagem profunda, pode ser entendida como uma classe de técnicas de aprendizado de máquina que explora muitas camadas de informação não lineares. Basicamente, *Deep Learning* trata do aprendizado de múltiplos níveis de representação e abstração que ajudam a construir sentido em dados, tais como imagens, sons e texto. Em suas diversas definições, duas características estão sempre presentes e permitem a representação de características em níveis de complexidade sucessivamente mais altos, permitindo uma abstração maior à cada nova camada, sendo um modelo que consiste de múltiplas camadas de processamento de informação não linear e também um método de aprendizado supervisionado, onde os dados inseridos já estão classificados e categorizados, e não supervisionado, onde a máquina classifica por conta própria os dados com base nos padrões encontrados (DENG; YU, 2014).

Porém o treinamento de uma rede neural utilizando *Deep Learning* pode ser bastante difícil e complexo, devido ao fato de que essas redes neurais utilizam mais camadas ocultas do que as redes neurais comuns. Considerando que em redes neurais comuns todos os neurônios de uma camada se conectam com todos os neurônios da próxima camada, a quantidade de parâmetros a serem ajustados durante as iterações no processo de treinamento do modelo é elevada, o que demandaria grande poder de processamento e um número de épocas de treinamento muito maior. Por isso utiliza-se de um outro conceito importantíssimo chamado de rede neural convolucional artificial, ou simplesmente rede convolucional. As redes convolucionais têm uma diferença fundamental em relação às redes neurais comuns. Nas redes convolucionais cada neurônio da camada de input equivale ao valor de um dos *pixels* da imagem e, diferentemente do que acontece nas redes neurais comuns, essa camada de input não é achatada, ou seja, não se transforma a matriz de 28x28 *pixels* em um vetor de 784 valores, mas preservam-se suas dimensões. Utilizando-se de uma técnica chamada de campos receptivos locais, ao invés de ligar cada um dos neurônios da primeira camada com todos os neurônios da segunda camada, é passado um filtro, ou *kernel*, com tamanho específico, por toda a imagem, que executa uma convolução na imagem. O valor resultante dessa convolução será transformado por uma função de ativação e passado para a próxima camada da rede neural.

Tomando como exemplo uma imagem de 28x28 *pixels*, a técnica dos campos receptivos locais consiste em selecionar um determinado campo dessa imagem com um tamanho específico, por exemplo um campo de 5x5 e filtrar esses valores para um único neurônio da próxima camada em um processo de convolução, em seguida, o mesmo filtro é deslocado para o lado na distância de um passo (valor a ser definido na criação do modelo, também chamado de *stride*), sendo este processo, repetido até que toda a imagem seja filtrada. No momento em que essa imagem é convolucionada são gerados mapas de característica, ou *feature maps*. Cada um desses mapas destaca uma característica específica da imagem e será usado como input da próxima camada. No caso da entrada de 28x28 *pixels*, utilizado-se um filtro de 5x5 e um passo de 1 neurônio, haveriam mapas de características na camada de convolução com o tamanho de 24x24 neurônios, devido à perda que acontece nas bordas dessa matriz pelo tamanho do filtro. Todos os neurônios de um determinado mapa de características receberiam um valor resultante de uma convolução com um único filtro de convolução, sendo esse, outro conceito importante das redes convolucionais, denominado de pesos

compartilhados, o qual permite que cada filtro produza um mapa de característica diferente que ressalte uma característica específica da imagem. Uma representação visual desse processo de convolução pode ser observado na Figura 1. Existe, ainda, uma técnica que previne a diminuição no tamanho das dimensões que acontece após uma convolução, chamada de *padding*, que consiste em aumentar o tamanho das dimensões da matriz acrescentando zeros ou a média dos *pixels* mais próximos nas bordas da matriz antes de aplicar o processo de convolução. O uso ou não, dessa técnica, é definido na construção do modelo e não foi usada neste último exemplo (NIELSEN, 2015).

Outro conceito importante é o agrupamento, ou *pooling*, também ilustrado na Figura 1.

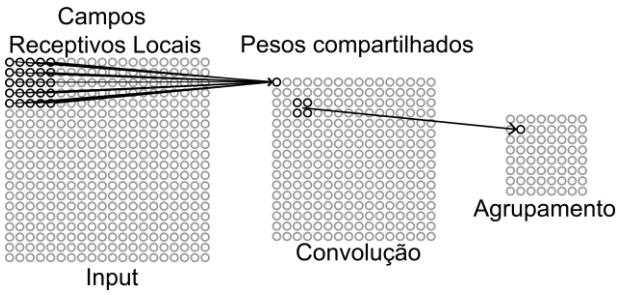

Figura 1 - *Campos receptivos locais, convolução e agrupamento. Fonte: Adaptado de Nielsen (2015)*

Geralmente, após uma camada de convolução, a próxima camada do modelo de rede convolucional será uma camada de agrupamento, cujo objetivo é gerar uma saída simplificada das características capturadas pela camada de convolução. Os mapas de características são dispostos em uma matriz, assim como os dados de entrada, dessa forma, no processo de agrupamento, geralmente usa-se um campo de 2x2 desses mapas e o valor dos campos são generalizados para um único neurônio da camada de agrupamento. Essa generalização pode ser feita de várias formas, porém a mais comum é conhecida como *max-pooling*, que consiste em utilizar o valor mais significativo do campo 2x2 e atribuir ao neurônio da camada de agrupamento. Diferentemente da convolução, os campos de agrupamento não repetem neurônios já agrupados, dando a ideia de que o passo para agrupamento seria igual ao tamanho do campo que, no caso do exemplo, seria de tamanho 2. Isso significa que a camada de agrupamento para o modelo sugerido teria 12x12 neurônios, levando em conta que a camada de convolução apresentou 24x24 *pixels* (NIELSEN, 2015).

Para implementar um modelo de redes neurais e outros algoritmos de aprendizado de máquina, dada sua complexidade e necessidade de alta performance, muitas ferramentas foram desenvolvidas nos últimos anos com o intuito de facilitarem o trabalho do pesquisador. Nesse quesito a linguagem de programação Python aparece com frequência devido a sua facilidade de implementação e leitura. No que diz respeito à performance, entretanto, essa linguagem não figura entre as melhores, mas existem ferramentas específicas que têm as partes mais importantes desenvolvidas em linguagens de alta performance, como a linguagem C, e possuem uma interface programável disponível em Python, garantindo a performance e facilidade de desenvolvimento desejada. Dentre as que mais se destacam e foram usadas no trabalho estão a biblioteca TensorFlow que é amplamente utilizada para pesquisa e melhoramento de algoritmos de aprendizado de máquina, abrangendo aplicações como reconhecimento de voz, visão computacional, robótica, processamento de linguagem natural, extração de informações geográficas, entre outras aplicações (ABADI et al., 2016); a biblioteca Keras que apresenta uma interface programável de alto nível utilizando estruturas e funcionalidades do TensorFlow; a biblioteca NumPy que fornece uma estrutura de dados para computação numérica (WALT; COLBERT; VAROQUAUX, 2011) e a Matplotlib, que possibilita a fácil criação de gráficos para visualização de dados (HUNTER, 2007).

O objetivo desse trabalho foi aplicar a técnica *Deep Learning* por meio de um modelo de rede convolucional e avaliar o desempenho desse modelo na análise e categorização de um conjunto de dados de imagem de defeitos de madeira. Como objetivo secundário, buscou-se entender qual é o impacto da alteração da quantidade de mapas na primeira camada convolucional e da dimensão dos campos receptivos locais.

2 MATERIAL E MÉTODOS

Neste trabalho foi empregada a versão 3.5.6 da linguagem de programação Python, além da utilização da biblioteca Keras, que simplificou bastante a construção, avaliação e validação de modelos de redes neurais e redes convolucionais (CHOLLET, 2015) e das bibliotecas TensorFlow (ABADI et al., 2016) e Numpy (WALT; COLBERT; VAROQUAUX, 2011), necessárias para o perfeito funcionamento da Keras. Essa dependência entre ferramentas e bibliotecas deve ser observada, ao optar-se por uma plataforma de desenvolvimento de modelos de redes neurais e aprendizado de máquina.

2.1 TENSORFLOW

TensorFlow é uma ferramenta de interface de desenvolvimento que tem como objetivo a elaboração e implementação de algoritmos de aprendizado de máquina, otimização e treinamento. É um sistema bastante flexível, podendo ser usado para expressar vários tipos de algoritmos, incluindo os de *Deep Learning*. Está em constante desenvolvimento e pode ser utilizada gratuitamente de forma comercial e em pesquisas científicas (ABADI et al., 2016). O TensorFlow possui funções para realizar a computação numérica de mapas de fluxo de dados, que consistem em nós e arestas onde cada nó representa a instanciação de uma operação e valores que fluem pelas arestas (ABADI; ISARD; MURRAY, 2017), trabalhando com matrizes multidimensionais dinâmicas que interagem com os nós do mapa, que em geral implementam modelos de operações matemáticas (TENSORFLOW, 2016). A versão usada foi a TensorFlow CPU 0.12.0.

2.2 NUMPY

A linguagem de programação Python, em sua configuração disponibilizada, não apresenta, originalmente, recursos para computação numérica de alta performance. Por esse motivo foi utilizada uma biblioteca com estruturas de dados desenvolvidas para a computação eficiente de matrizes (ou *arrays*) numéricas destinada a linguagem Python. A sua distribuição e utilização é gratuita, abrangendo, além da estrutura de matrizes, muito utilizada em reconhecimento de padrões em imagens, um conjunto de funções matemáticas para manipulação dessas matrizes. Um *array* em NumPy é uma coleção multidimensional de elementos, podendo ter até 32 dimensões e conter diferentes tipos e combinações de elementos (WALT; COLBERT; VAROQUAUX, 2011). A versão usada foi a NumPy 1.15.2.

2.3 KERAS

A Keras é uma biblioteca, também de código aberto, para aplicação de redes neurais, desenvolvida em linguagem Python, com um conjunto de funções de alto nível trabalhando com recursos do TensorFlow e do Theano. O Theano é uma biblioteca mais antiga em comparação com o Tensorflow e possui recursos parecidos, porém não é muito utilizado nos estudos mais recentes devido a algumas limitações superadas pelo seu sucessor (AL-RFOU et al., 2016). A biblioteca Keras foi desenvolvida com foco na experimentação rápida, fazendo com que seja relativamente rápido produzir resultados a partir do momento da geração da ideia, imprimindo mais agilidade em pesquisas onde a biblioteca possa ser aplicada. Uma de suas principais características é permitir uma prototipagem descomplicada e eficiente (CHOLLET, 2015). A versão usada foi a Keras 1.2.2.

2.4 MATPLOTLIB

A biblioteca baseada em Python é chamada de Matplotlib, sendo muito utilizada em pesquisas e publicações. Tem como objetivo gerar figuras e gráficos em duas dimensões que ilustrem, de forma adequada, um conjunto de informações e sua evolução. Emprega-se essa ferramenta na geração de gráficos de exibição de dados obtidos durante os treinamentos e avaliações de modelos de redes neurais de vários tipos, incluindo as redes neurais convolucionais. A ferramenta possibilita gerar, entre outros tipos de representações gráficas, histogramas, planos, espectros de força, gráficos de barra e gráficos de erro (HUNTER, 2007). A versão usada foi Matplotlib 3.0.0

2.5 CONJUNTO DE DADOS

As imagens utilizadas neste trabalho foram originadas a partir de um conjunto de 408 imagens de tábuas de madeira serrada de Pinus, com tamanhos variados, sendo 324 imagens coletadas por Gomes (2013) e 84 imagens por Rall (2010). Posteriormente, em outro estudo, essas 408 imagens foram subdivididas em blocos menores, com resoluções de 128x128, 64x64 e 32x32 *pixels*, originando milhares de imagens (ALMEIDA, 2014). Optou-se por trabalhar com as imagens de 32x32 *pixels*, totalizando um conjunto de 440 imagens de madeira, dividido em duas categorias, sendo uma de imagens que apresentam porção de madeiras com defeitos e outra de imagens de madeiras sem defeitos. Desse total, de 440 imagens, 293 foram reservadas para o treinamento do modelo convolucional e 147 para teste do modelo.

Os conjuntos de imagens de treinamento e teste foram divididos em um grupo com blocos de imagens que apresentavam defeitos e outro, sem defeitos. Pela Figura 2, observa-se ambas as situações. No caso do conjunto de 293 imagens de treinamento, 160 imagens são exemplares de madeira sem defeito e 133 imagens são exemplares de madeira com defeito. Para o conjunto de 147 imagens de teste, 67 são exemplares de imagens de madeira com defeito, sendo as 80 restantes, de exemplares de madeiras sem defeito.

As imagens foram gravadas no padrão de cores RGB (três bandas de cores: vermelha, verde e azul), composto por 3 canais de cores, fator considerado para a construção do modelo convolucional. A opção pelas imagens com menor resolução se justifica no fato de que, quanto menor a resolução, menos dados serão processados durante o treinamento, o que consequentemente agiliza o processo de testes do modelo de redes neurais construído.

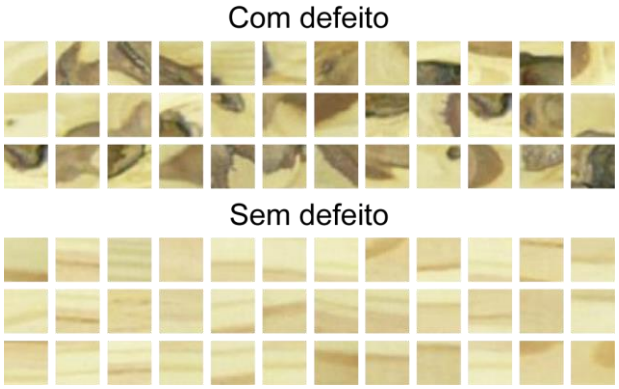

Figura 2 *- Exemplares de imagens de 32x32 pixels do Conjunto de dados de madeiras. Fonte: Gomes (2013) e Almeida (2014)*

2.6 MODELO DE REDE CONVOLUCIONAL

Para a construção do modelo foi necessário analisar as características das imagens trabalhadas. Cada imagem do conjunto de dados possui uma dimensão de 32x32 *pixels*, com o padrão de cores RGB. Dessa maneira, o modelo de treinamento foi definido com 3 unidades de entrada de 32x32 neurônios, totalizando 3072 *inputs*, sendo 1024 unidades por banda de cor. Todas as imagens do conjunto de dados foram processadas pela rede convolucional em épocas. Uma época de treinamento consiste em cada ciclo de processamento completo de todas as imagens do conjunto.

O modelo utilizado foi adaptado de um outro modelo denominado por Brownlee (2016a), como *larger_model*, o qual obteve, como resultado, 0.89% de erro no processamento de um conjunto de dados voltado para estudos conhecido como *Mnist*. Basicamente, as diferenças entre esse modelo (*larger_model)* e o utilizado

foram, a quantidade de mapas de característica e a dimensão dos campos receptivos locais na primeira camada de convolução. Como feito no modelo original, para que os resultados pudessem ser reproduzidos de forma aproximada, os pesos e vieses do modelo foram gerados de forma pseudoaleatórias. Isso se tornou possível com a utilização de uma variável *seed* definida no início do algoritmo como sendo igual a 7 (sete). A função que gera os números aleatórios faz uso dessa variável, gerando os mesmos números pseudoaleatórios sempre que for usado o mesmo valor em *seed*.

Uma das camadas do modelo realiza o chamado *dropout*, que é uma solução que propõe o descarte de uma certa porcentagem dos *outputs* da camada anterior para próxima camada do modelo, visando prevenir o fenômeno do *overfitting* que, por sua vez, acontece quando o modelo começa a utilizar muitos parâmetros aprendidos durante o treinamento para a categorização, fazendo com que os resultados obtidos nas amostras do treinamento sejam muito bons, porém, quando se tenta analisar um conjunto de imagens de distinto ao conjunto de treinamento, um conjunto de teste por exemplo, o modelo passa a produzir resultados ruins, o que diminui sua praticidade. O *dropout* faz com que os resultados obtidos no teste do modelo de rede convolucional sejam melhores (SRIVASTAVA et al., 2014).

Outras camadas importantes do modelo são as camadas de achatamento ou *flatten* e as camadas finais conectadas aplicando-se a estrutura do tipo todos-todos, também conhecidas como camadas densamente conectadas, semelhante a redes neurais artificiais simples. A camada de achatamento transforma a estrutura dos *outputs* bidimensionais, que vêm das camadas anteriores, em uma estrutura linear. Essa estrutura linear pode ter em sua sequência outra camada com estrutura linear ou a camada de *output* do modelo, onde o resultado é obtido.

A arquitetura do modelo é dependente do equipamento de hardware disponível, pois qualquer alteração na quantidade de camadas, épocas de treinamento e demais detalhes da arquitetura, interferem no tempo necessário para treinar o modelo podendo, dependendo dos valores definidos, inviabilizar o processo.

Na topologia, tem-se um modelo sequencial, onde a camada 1 é uma camada de convolução com 30 mapas de característica, campos receptivos locais definidos com dimensão de 5x5 e o *input* tem o formato de 32x32 *pixels* e 3 canais de cores; a camada 2 é um agrupamento (*pooling*) usando um campo de dimensão 2x2; a camada 3 é uma convolução com 15 mapas de características e campos receptivos locais de 3x3; a camada 4 é um agrupamento com um campo de 2x2; a camada 5 executa um *dropout* de 20% da camada 4 para gerar a sexta camada; a camada 6 é uma camada de achatamento (*flatten*); a camada 7 é uma camada de 128 neurônios densamente conectada com a camada 8 (conexão todos-todos); a camada 8 é uma camada de 50 neurônios densamente conectada com a camada 9; e a camada 9 é a camada de *output* (2 neurônios) que calcula a probabilidade da imagem pertencer à cada uma das duas classes, onde o maior valor será o resultado considerado. Esse modelo foi treinado e testado em 50 épocas. A Figura 3 exemplifica a estrutura do modelo.

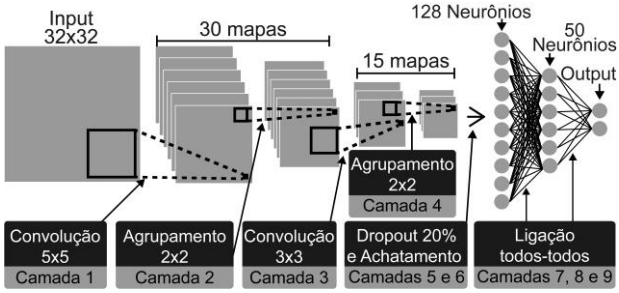

Figura 3 - *Modelo de rede convolucional utilizado*

Para que se pudesse entender como os elementos das camadas convolucionais afetam o resultado final, foram definidas pequenas variações na camada 1 do modelo, com os campos receptivos locais fixos numa dimensão de 5x5, variando-se as quantidades dos mapas de característica, nos valores de 20, 30 e 40. Após três execuções do modelo para cada uma das variações nos valores do mapa de característica, foi feito o teste no melhor modelo variando a dimensão dos campos receptivos locais nos valores de 4x4 e 6x6, também com três execuções para cada um desses modelos, ressaltando-se que já havia sido executado com o valor de 5x5. Dessa maneira, foi possível avaliar o desempenho da *Deep Learning* na categorização de dados de madeira e entender os mecanismos das redes convolucionais.

Os resultados obtidos foram exibidos por meio de tabelas e gráficos gerados pela biblioteca Matplotlib (BROWNLEE, 2016b).

3 RESULTADOS E DISCUSSÕES

Durante a execução dos modelos propostos foram obtidos resultados bastante satisfatórios. Inicialmente foram analisados três modelos, o primeiro com 40 mapas de característica, o segundo com 30 e o terceiro, com 20. Todos os modelos tiveram como dimensão dos campos receptivos locais, 5x5. Conforme observa-se Tabela 1, foram observadas variações consideráveis nos resultados para esses modelos.

Tabela 1 - *Resultado dos modelos com quantidade de mapas de característica variados e campos receptivos locais de 5x5*

Qtd. de mapas	Percentual de Erro			Tempo médio (min:seg)
	1ª exec.	2ª exec.	3ª exec.	
40	10,88%	11,56%	7,48%	**1:51,96**
30	14,97%	5,44%	8,84%	**1:21,28**
20	6,80%	6,12%	5,44%	**1:01,07**

Os resultados mostraram que, para o conjunto de dados de imagens de madeira utilizado, o aumento da quantidade de mapas de característica na primeira camada gerou

resultados menos satisfatórios. Isso ocorreu, provavelmente, devido à pequena quantidade de características que são realmente úteis e relevantes para a categorização correta. O aumento desses mapas de característica forçou o modelo a analisar características não relevantes para a categorização entre imagens de madeiras com ou sem defeito, fazendo com que o modelo aprendesse erroneamente quais características eram importantes, gerando uma porcentagem maior de erros.

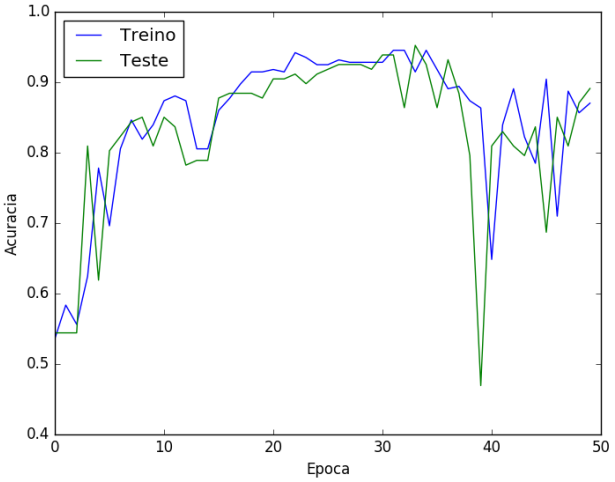

Figura 4 - *Gráfico de acurácia em cada época para o modelo de 40 mapas de características.*

Pode-se perceber a inconsistência gerada devido à grande variação nos resultados com mais de 20 mapas de característica. A Figura 4 apresenta os gráficos gerados a partir dos resultados obtidos na primeira execução do modelo com 40 mapas de característica, tendo como métrica a acurácia em relação às épocas. É interessante notar que o modelo apresenta resultados ruins pouco depois da trigésima época, indicando que, para modelos com muitos mapas de característica, seria interessante diminuir a quantidade de épocas, para a obtenção de resultados melhores.

Como pode ser observado na Tabela 1, o modelo com 20 mapas de característica foi o que apresentou os melhores resultados nas três execuções de amostra, sendo o escolhido para a realização dos novos testes, onde foram variadas as dimensões dos campos receptivos locais para 4x4 e 6x6, considerando-se o resultado já obtido para o dimensionamento 5x5. Os resultados obtidos podem ser observados na Tabela 2. O resultado para os campos de 5x5 foram repetidos para facilitar a visualização.

Tabela 2 - *Resultados dos modelos de 20 mapas de característica e dimensão dos campos receptivos locais variados*

Dimensão	Percentual de Erro			Tempo médio
	1ª exec.	2ª exec.	3ª exec.	(min:seg)
4x4	6,12%	6,12%	5,44%	**1:00,28**
5x5	6,80%	6,12%	5,44%	**1:01,07**
6x6	9,52%	5,44%	18,37%	**1:01,97**

Como era esperado, o aumento na dimensão dos campos receptivos locais proporcionou resultados piores para o conjunto de imagens usado. Isso não é uma regra para todos os conjuntos, porém, as imagens utilizadas eram de 32x32, fazendo com que a região captada nos campos receptivos não apontasse adequadamente as características desejas.

Porém, pelo mesmo motivo, obtiveram-se resultados ainda melhores quando esses campos foram diminuídos para 4x4 nas suas dimensões. Como pode-se constatar na Figura 5, os resultados variaram menos, aumentando a confiabilidade do modelo.

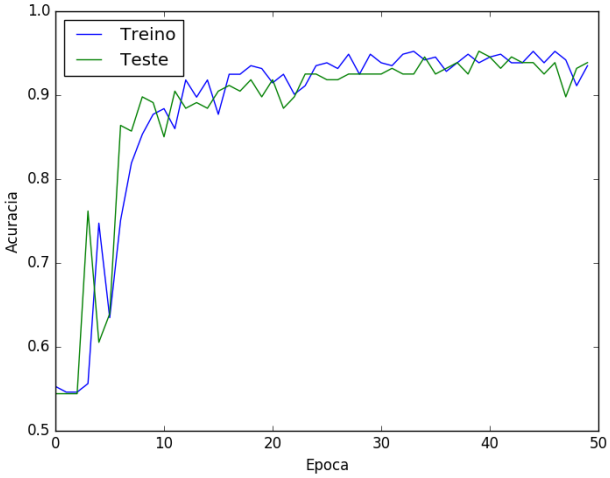

Figura 5 - *Acurácia do modelo de 20 mapas com campos receptivos locais de 4x4*

Outro fator observado foi que nos modelos treinados, quanto maior a quantidade de mapas de característica sendo utilizados pelo modelo, maior foi o tempo necessário para o processamento das imagens. Esses e os outros resultados já descritos apontaram para o modelo onde se tem 20 mapas de característica na primeira camada e campos receptivos locais com dimensão de 4x4.

Como observado, a técnica de aprendizado de máquina *Deep Learning* apresenta resultados promissores, tendo como melhor deles o valor de 5,44% de erro na classificação de imagens de madeira com e sem defeito (8 imagens classificadas erroneamente em um total de 147 imagens de teste). Rall (2010), utilizando *software* desenvolvido para classificação de cinco categorias de tábuas de madeira e métodos clássicos de processamento de imagens e extração de características, observou erro de classificação das tábuas de 9,5%. Vale ressaltar que essa porcentagem de erro mais alta experimentada por Rall pode ser devido a maior complexidade do problema que seu *software* visava solucionar e também pelo tamanho das imagens analisadas. Almeida (2014), faz uma extensa análise com diferentes modelos de aprendizado de máquina, coletando diversos resultados, onde o melhor deles apresentou 3,12% de erro na classificação de seis classes distintas de defeitos em imagens de madeira de 32x32 pixels utilizando a técnica SVM.

4 CONCLUSÃO

A utilização da técnica *Deep Learning* apresenta resultados interessantes, mesmo usando um modelo de redes convolucionais relativamente simples e não otimizado especificamente para a classificação das imagens de madeira. Isso viabiliza a continuidade do estudo para um modelo mais otimizado e que categorize o defeito das madeiras.

O sucesso do modelo utilizado foi possível devido às características das redes convolucionais, como os campos receptivos locais, o agrupamento, o compartilhamento de pesos entre um mesmo mapa de característica, além da organização dos neurônios em formato bidimensional (ou não linear). Tais características e a correta aplicação das mesmas possibilitaram, de acordo com os resultados, um bom processamento das imagens 2D.

Para cada conjunto de dados existe uma configuração de topologia mais adequada, tendo em vista que o mesmo modelo teve uma performance muito boa na classificação de dígitos e um pouco pior na classificação de madeiras com e sem defeito. A alteração de valores nos mapas de característica e nos campos receptivos locais, trazendo resultados diferentes, também evidenciaram tal fato. Outro ponto foi que os modelos apresentam tempos de processamento diferentes e, dependendo do caso, um modelo pode apresentar um resultado significativamente bom e consistente, porém, o tempo necessário para o processamento pode inviabilizar sua aplicação.

Outra característica relevante da *Deep Learning* foi que essa técnica possibilita que estudiosos do aprendizado de máquina possam contribuir com pesquisas de outras áreas de atuação, sem ter, necessariamente, que especializarem-se naquela área, viabilizando a formação de equipes multidisciplinares. Como no exemplo do presente trabalho de pesquisa, um praticante do aprendizado de máquina, aplicando *Deep Learning*, não precisa necessariamente saber quais são as características específicas dos defeitos encontrados nas tábuas de madeira de *Pinus* para o seu emprego nas soluções propostas e necessárias para o seu desenvolvimento.

5 REFERÊNCIAS

ABADI, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. **Arxiv**, p. 1-19, 16 mar. 2016. Disponível em: <https://arxiv.org/pdf/1603.04467v2.pdf>. Acesso em: 19 nov. 2016.

ABADI, M.; ISARD, M.; MURRAY, D. G. A Computational Model for TensorFlow: An Introduction. In: ACM SIGPLAN INTERNATIONAL WORKSHOP ON MACHINE LEARNING AND PROGRAMMING LANGUAGES, 1., 2017, Barcelona. **Proceedings.** New York: ACM, 2017. p. 1-7. Disponível em: <http://doi.acm.org/10.1145/3088525.3088527>. Acesso em: 24 set. 2018.

ALMEIDA, O. C. P. **Classificação de tábuas de madeira usando processamento de imagens digitais e aprendizado de máquina**. 2014. 105 f. Tese (Doutorado em Agronomia - Energia na Agricultura) -Faculdade de Ciências Agronômicas, Universidade Estadual Paulista, Botucatu. Disponível em: <http://hdl.handle.net/11449/115579>. Acesso em: 27 nov. 2018.

AL-RFOU, R. et al. Theano: A Python framework for fast computation of mathematical expressions. **Arxiv,** p. 1-19, 9 maio 2016. Disponível em: <https://arxiv.org/pdf/1605.02688.pdf>. Acesso em: 20 nov. 2016.

BROWNLEE, J. Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras. Vermont: Machine Learning Mastery, 2016a. Disponível em: <http://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/>. Acesso em: 24 nov. 2016.

BROWNLEE, J. Display Deep Learning Model Training History in Keras. Vermont: Machine Learning Mastery 2016b. Disponível em: <http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/>. Acesso em: 24 nov. 2016.

CHOLLET, F. Keras: Deep Learning library for Theano and TensorFlow**.** [S.l.]: GitHub, 2015. Disponível em: <https://github.com/fchollet/keras>. Acesso em: 20 nov. 2016.

DENG, L.; YU, D. Introduction: Definitions and Background. In: DENG, L.; YU, D. **Deep Learning Methods and Applications**. Boston: Now Publishers, 2014. Chap. 1, p. 199-200.

GOMES, R. C. **Desenvolvimento de uma Base de Dados de Imagens Digitais De Madeira Serrada de Coníferas**. 2013. 93 f. Dissertação (Mestrado em Agronomia - Energia na Agricultura) - Faculdade de Ciências Agronômicas, Universidade Estadual Paulista, Botucatu. Disponível em: <http://hdl.handle.net/11449/90628>. Acesso em: 27 nov. 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Introduction: Historical Trends in Deep Learning. In: GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A**. Deep Learning**. Cambridge: Mit Press, 2017. p. 12-18. Disponível em: <https://www.deeplearningbook.org>. Acesso em: 24 set. 2018.

HASSOUN, M. H. Threshold Gates. In: HASSOUN, M. H. **Fundamentals of Artificial Neural Networks**. London: The Mit Press, 1995. Chap. 1, p. 1.

HUNTER, J. D. Matplotlib: A 2D Graphics Environment. **Computing In Science & Engineering,** New York, v. 9, n. 3, p. 90-95, 18 jun. 2007.

Disponível em: <http://dx.doi.org/10.1109/MCSE.2007.55>. Acesso em: 20 nov. 2016.

KLINE, D. E.; SURAK, C.; ARAMAN, P. A. Automated hardwood lumber grading utilizing a multiple sensor machine vision technology. **Computers and Electronics in Agriculture**, Virginia, v. 41, n. 1/3, p. 139-155, 2003. Disponível em:<http://www.sciencedirect.com/science/article/pii/S0168169903000486>. Acesso em: 28 nov. 2018.

NIELSEN, M. A. Using neural nets to recognize handwritten digits. In: NIELSEN, M. A. **Neural Networks and Deep Learning**. San Francisco: Determination Press, 2015. Chap. 1. Disponível em: <http://neuralnetworksanddeeplearning.com/chap1.html>. Acesso em: 20 out. 2016.

RALL, R. **Processamento de imagens digitais para detecção e quantificação de defeitos na madeira serrada de coníferas de reflorestamento de uso não estrutural**. 2010. 123 f. Tese (Doutorado em Agronomia - Energia na Agricultura) -Faculdade de Ciências Agronômicas, Universidade Estadual Paulista, Botucatu. Disponível em: <http://hdl.handle.net/11449/101882>. Acesso em: 27 nov. 2018.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. Introduction. In: SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding Machine Learning:** From Theory to Algorithms. New York: Cambridge University Press, 2014. Chap. 1, p. 1-6.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal Of Machine Learning Research,** Toronto, p. 1929-1958. jun. 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>. Acesso em: 24 nov. 2016.

TENSORFLOW. **TensorFlow is an Open Source Software Library for Machine Intelligence.** Mountain View, 2015. Disponível em: <https://www.tensorflow.org/>. Acesso em: 2 nov. 2016.

WALT, S.; COLBERT, S. C.; VAROQUAUX, G. The NumPy Array: a structure for efficient numerical computation. **Computing In Science & Engineering**, New York, v. 13, n. 2, p. 22-30, 14 mar. 2011. Disponível em: <http://dx.doi.org/10.1109/MCSE.2011.37>. Acesso em: 24 nov. 2016.