



UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Botucatu



ROGER CRISTHIAN GOMES

**APLICAÇÃO DE *DEEP LEARNING* NA CLASSIFICAÇÃO DE TÁBUAS DE
MADEIRA POR MEIO DE ANÁLISE DE IMAGENS DIGITAIS**

Botucatu

2019

ROGER CRISTHIAN GOMES

**APLICAÇÃO DE *DEEP LEARNING* NA CLASSIFICAÇÃO DE TÁBUAS DE
MADEIRA POR MEIO DE ANÁLISE DE IMAGENS DIGITAIS**

Tese apresentada à Faculdade de Ciências Agronômicas da Unesp Câmpus de Botucatu, para obtenção do título de Doutor em Agronomia (Energia na Agricultura).

Orientador: Prof.Dr. Adriano Wagner Ballarin

Coorientador: Prof.Dr. Osvaldo César Pinheiro de Almeida

Botucatu

2019

G633a Gomes, Roger Cristhiam
 Aplicação de deep learning na classificação de tábuas
de madeira por meio da análise de imagens digitais /
Roger Cristhiam Gomes. -- Botucatu, 2019
 98 p.

 Tese (doutorado) - Universidade Estadual Paulista
(Unesp), Faculdade de Ciências Agrônomicas, Botucatu
Orientador: Adriano Wagner Ballarin
Coorientador: Osvaldo César Pinheiro de Almeida

 1. Aprendizado profundo. 2. Automatização. 3. Pinus.
4. Redes Neurais Convolucionais. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências Agrônomicas, Botucatu. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

CERTIFICADO DE APROVAÇÃO

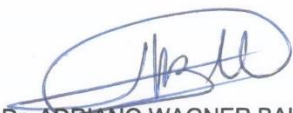
Título: **“APLICAÇÃO DE *DEEP LEARNING* NA CLASSIFICAÇÃO DE TÁBUAS DE MADEIRA POR MEIO DE ANÁLISE DE IMAGENS DIGITAIS”**

AUTOR: ROGER CRISTHIAN GOMES

ORIENTADOR: ADRIANO WAGNER BALLARIN

COORDENADOR: OSVALDO CESAR PINHEIRO DE ALMEIDA

Aprovado como parte das exigências para obtenção do Título de Doutor em AGRONOMIA (ENERGIA NA AGRICULTURA), pela Comissão Examinadora:



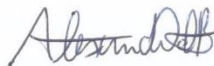
Prof. Dr. ADRIANO WAGNER BALLARIN
Engenharia Rural / Faculdade de Ciências Agrônômicas de Botucatu



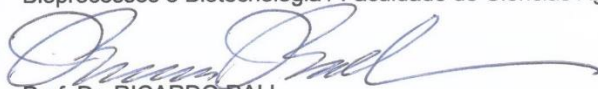
Prof. Dr. DIEGO AUGUSTO DE CAMPOS MORAES
Análise e Desenvolvimento de Sistemas / Faculdade Eduvale de Avaré



Prof. Dr. CARLOS ROBERTO PEREIRA PADOVANI
Informática / Faculdade de Tecnologia de Botucatu



Prof. Dr. ALEXANDRE DAL PAI
Bioprocessos e Biotecnologia / Faculdade de Ciências Agrônômicas de Botucatu - UNESP



Prof. Dr. RICARDO RALL
Tecnologia / Faculdade de Tecnologia de Botucatu

Botucatu, 31 de maio de 2019.

*Ao meu pai,
em memória!*

AGRADECIMENTOS

À Deus.

Ao querido professor Adriano Wagner Ballarin, pela amizade, ensinamentos, orientações e, principalmente, pelo apoio e incentivo nos momentos difíceis que passei nesse período.

Ao amigo Osvaldo Cesar, pelo apoio e orientações.

À minha mãe, que sempre me agraciou com palavras de conforto, fé e encorajamento.

À minha esposa Ângela, companheira em todos os momentos.

RESUMO

O setor madeireiro e toda sua cadeia produtiva possuem grande força e importância para a economia brasileira, representando 1,5% do produto interno bruto nacional em 2016. Toda madeira serrada deveria, idealmente, ser submetida a uma classificação para definição mais precisa do seu destino e justa de seu valor comercial. Quando essa madeira serrada é destinada ao exterior, a classificação é, na maioria das vezes, obrigatória. Nas serrarias do país que em sua maioria são pequenas e pouco automatizadas, a classificação é normalmente feita por visão humana, ou seja, um profissional faz a análise visual de cada peça e a classifica segundo algum critério. Como em todo processo que envolve capacidade humana, o erro é inerente e, nesse caso, elevado, em torno de 52%, segundo a literatura. Dada a importância do setor, a demanda de matéria prima e a necessidade crescente dessa classificação, é extremamente justificável que esse processo seja aperfeiçoado. A alternativa é a automatização, visando sobretudo o aumento no acerto dessa classificação. O objetivo deste trabalho foi desenvolver um modelo de redes neurais artificiais usando *Deep Learning* (DL) para a classificação automatizada de madeiras serradas de Pinus, seguindo as recomendações das normas da ABNT. O modelo aplicou Redes Neurais Convolucionais (*Convolutional Neural Network* - CNN), técnica muito estudada recentemente e promissora em diversas áreas, principalmente no processamento de imagens digitais e visão de máquina. Foram experimentados vários modelos, sendo o de melhor performance com acurácia de 97,50%. Comparando com outras pesquisas na mesma área concluiu-se que DL com CNN produz resultados aceitáveis na classificação de tábuas, mesmo com poucas imagens (284), diferença na variedade do Pinus (*elliottii* e *taeda*) e apresentação (madeira verde ou seca, aplainada ou não).

Palavras-chave: Aprendizado profundo. Automatização. Pinus. Redes Neurais Convolucionais.

ABSTRACT

The timber sector and its entire production chain have great strength and importance for the Brazilian economy, representing 1.5% of the national gross domestic product in 2016. All lumber should ideally be subjected to a classification for a more precise definition of its destination and fairness of its commercial value. When this lumber is destined to the outside, classification is, in most cases, mandatory. In the country sawmills that are mostly small and little automated, the classification is usually done by human vision, that is, a professional makes the visual analysis of each piece and classifies it according to some criterion. As in any process involving human capacity, the error is inherent and, in this case, high, around 52%, according to the literature. Given the importance of the industry, the demand for raw materials and the growing need for such classification, it is extremely justifiable that this process is improved. The alternative is automation, aiming in particular to increase the accuracy of this classification. The objective of this work was to develop a model of artificial neural networks using Deep Learning (DL) for the automated classification of *Pinus* sawn timber, following the recommendations of ABNT standards. The model applied Convolutional Neural Network (CNN), a very recently studied and promising technique in several areas, mainly in digital image processing and machine vision. Several models were tried, being the one of better performance with accuracy of 97.50%. It was concluded that DL with CNN produces acceptable results in the classification of boards, even with few images (284), difference in the *Pinus* variety (*elliottii* and *taeda*) and presentation (green or dry wood, planed or not).

Keywords: Automation. Convolutional Neural Networks. Deep learning. *Pinus*.

LISTA DE ILUSTRAÇÕES

Figura 1 - Área de árvores plantadas em milhões de hectares	15
Figura 2 - Áreas de árvores plantadas no brasil por estado e por gênero, 2016.....	16
Figura 3 - a) Defeito caracterizado nó firme; b) Defeito caracterizado nó vazado.....	18
Figura 4 - Defeito caracterizado como nó cariado.....	18
Figura 5 - Defeito caracterizado como bolsa de resina.	19
Figura 6 - Defeito caracterizado como medula.....	19
Figura 7 - Presença de nó firme em tábua de classe Super	19
Figura 8 - Presença de nó vazado em tábua de terceira classe.....	20
Figura 9 - Exemplos de medição de nós ou grupo de nós	20
Figura 10 - Exemplos de medição de nós	22
Figura 11 - Representação do neurônio artificial McCulloch e Pitts em 1943	25
Figura 12 - Modelo atual de neurônio artificial.....	25
Figura 13 - Modelo de neurônio artificial MLP	27
Figura 14 - Esquema de rede neural convolucional	28
Figura 15 - Recurso <i>padding</i> , criação de borda dupla com zeros	29
Figura 16 - Camadas de convolução e subamostragem	30
Figura 17 - Parâmetros no processo de convolução	31
Figura 18 - Comportamento das funções de ativação.....	32
Figura 19 - Tábua completa e blocos de imagem extraídos da tábua.....	42
Figura 20 - Detalhe de algumas tábuas desdobradas.....	43
Figura 21 - Detalhe de uma tábua após o refilamento	44
Figura 22 - Tábua após o aplainamento e secagem	44
Figura 23 - Imagem da tábua após recorte	44
Figura 24 - Imagem da tábua de <i>Pinus taeda</i>	46
Figura 25 - Imagens de blocos e suas dimensões	47
Figura 26 - Imagens de blocos com 32x32 pixels	47
Figura 27- Exemplos das imagens de 128x128 pixels	48
Figura 28- Classificação e rótulos das imagens do Grupo 2	48
Figura 29 - Arquitetura modelo rede convolucional primeira abordagem	54
Figura 30 - Modelo de rede convolucional utilizado - segunda abordagem	58
Figura 31 - Fluxo de camadas do modelo	59
Figura 32 - Imagem após <i>custom_augmentation</i> técnica <i>crop 100x100</i>	62

Figura 33 - Arquitetura do modelo Final.....	67
Figura 34 - Formato de saída da função de predição	69
Figura 35 - Acurácia em cada época do modelo de 40 mapas e filtro de 5x5	71
Figura 36 - Acurácia em cada etapa do modelo de 20 mapas e filtro de 4x4	72
Figura 37 - Evolução do <i>loss</i> segunda abordagem - modelo inicial.....	74
Figura 38 - Evolução da acurácia na segunda abordagem - modelo inicial.....	75
Figura 39 - Evolução da acurácia no experimento 2.....	77
Figura 40 - Evolução do <i>loss</i> no experimento 2.....	77
Figura 41 - Valores de saída do modelo, detalhe dos valores próximos	78
Figura 42 - Valores de predição muito próximos	78
Figura 43 - Evolução do <i>loss</i> no melhor dos experimentos rodados no PC.....	80
Figura 44 - Evolução da acurácia no melhor dos experimentos rodados no PC	80
Figura 45 - Comportamento do <i>loss</i> - modelo Adaptado	82
Figura 46 - Comportamento da acurácia - modelo Adaptado	82
Figura 47 - Comportamento do <i>loss</i> - modelo Ajustado.....	85
Figura 48 - Comportamento da acurácia - modelo Ajustado.....	85
Figura 49 - Comportamento do <i>loss</i> - modelo Simplificado	87
Figura 50 - Comportamento da acurácia - modelo Simplificado	87
Figura 51 - Comportamento do <i>loss</i> - modelo Final.....	89
Figura 52 - Comportamento da acurácia - modelo Final.....	89

LISTA DE TABELAS

Tabela 1 - Dimensões padronizadas madeira serrada coníferas reflorestamento	17
Tabela 2 - Classificação madeira serrada coníferas provenientes reflorestamento ..	21
Tabela 3 - Trabalhos realizados na área.....	40
Tabela 4 - Produtos disponíveis no mercado e seus recursos	41
Tabela 5 - Classificação das tábuas de <i>Pinus elliotti</i>	45
Tabela 6 - Classificação das tábuas de <i>Pinus taeda</i>	46
Tabela 7 - Parâmetros das classes das imagens do Grupo 2.....	49
Tabela 8 - Distribuição das imagens de tábua completa, por classe.....	50
Tabela 9 - Nomenclatura e características dos conjuntos de imagens	50
Tabela 10 - Divisão das imagens do Grupo 1	53
Tabela 11 - Quantidade de imagens por classe para treino e teste, Grupo 2	56
Tabela 12 - Divisão das imagens do Grupo 3	60
Tabela 13 - Arquitetura do modelo Adaptado.....	61
Tabela 14 - Arquitetura do Modelo Ajustado com melhores resultados	63
Tabela 15 - Hiperparâmetros e valores definidos experimento modelo Ajustado.....	64
Tabela 16 - Arquitetura da melhor variação do modelo Simplificado	65
Tabela 17 - Configuração do modelo Final	68
Tabela 18 - Resultado dos modelos com filtro de 5x5.....	70
Tabela 19 - Resultados dos modelos 20 mapas característica e dimensão filtros	71
Tabela 20 - Comparativo de classificadores para 2 classes	73
Tabela 21 - Resultados do modelo inicial.....	73
Tabela 22 - Matriz de confusão da classificação 128x128, pelo modelo inicial.....	75
Tabela 23 - Melhores resultados do modelo final, para o conjunto de teste	76
Tabela 24 - Matriz de confusão da classificação 128x128, pelo modelo final	79
Tabela 25 - Melhor resultado dos modelos rodados no PC	80
Tabela 26 - Resultados de trabalhos na mesma área.....	81
Tabela 27 - Melhor <i>loss</i> do modelo Adaptado	81
Tabela 28 - Melhor acurácia do modelo Adaptado.....	81
Tabela 29 - Matriz de confusão modelo Adaptado	83
Tabela 30 - Melhores resultados do modelo Adaptado, 200 imagens balanceadas ..	83

Tabela 31 - Comparativo do modelo Adaptado	83
Tabela 32 - Melhor <i>loss</i> do modelo aplicando <i>image_augmentation</i>	84
Tabela 33 - Melhor acurácia do modelo aplicando <i>image_augmentation</i>	84
Tabela 34 - Matriz de confusão modelo Ajustado.....	85
Tabela 35 - Melhores resultados para <i>loss</i> , modelo simplificado.....	86
Tabela 36 - Melhores resultados para acurácia, modelo simplificado	87
Tabela 37 - Matriz de confusão, modelo Simplificado	88
Tabela 38 - Melhores resultados do modelo para <i>loss</i>	88
Tabela 39 - Melhores resultados do modelo para acurácia	89
Tabela 40 - Matriz de confusão, modelo Final.....	90
Tabela 41 - Comparativo dos resultados de trabalhos na mesma área.....	90

LISTA DE ABREVIATURAS E SIGLAS

ABIMCI	Associação Brasileira da Indústria de Madeira Processada Mecanicamente
ABNT	Associação Brasileira de Normas Técnicas
ABPM	Associação Brasileira dos Produtores de Madeira
AE	Auto Encoders
AG	Algoritmo Genético
AI	Artificial Intelligence
ANN	Artificial Neural Networks
BMP	Bitmap
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DBN	Deep Belief Networks
DL	Deep Learning
DT	Decision Trees
GIF	Graphics Interchange Format
GIMP	Gnu Image Manipulation Program
GPU	Graphics Processing Unit
IA	Inteligência Artificial
IBÁ	Indústria Brasileira da Árvore
JPEG	Join Photographics Experts Group
KNN	K-Nearest Neighbor
LED	Light Emissor Diode
ML	Machine Learning
PC	Personal Computer
PIB	Produto Interno Bruto
PMVA	Produto de Maior Valor Agregado
RBF	Radial-Basis Function
RGB	Red Green Blue
RNA	Redes Neurais Artificiais
SVM	Support Vector Machine
TIFF	Tagged Image File Format

SUMÁRIO

1	INTRODUÇÃO	11
2	REVISÃO DE LITERATURA	15
2.1	Madeira serrada no Brasil	15
2.2	Defeitos e classificação de tábuas de coníferas	17
2.3	Aprendizado de máquina	22
2.3.1	Rede Neural Artificial - RNA	24
2.3.2	Deep Learning e Redes Neurais Convolucionais	27
2.3.2.1	<i>Extensão da imagem ou Padding</i>	29
2.3.2.2	<i>Subamostragem (pooling)</i>	30
2.3.2.3	<i>Função de ativação por retificação linear</i>	31
2.3.2.4	<i>Regularização L2</i>	32
2.3.2.5	<i>Métricas de qualidade do modelo</i>	32
2.3.2.6	<i>Algoritmos de otimização</i>	33
2.4	Ferramentas e bibliotecas aplicadas a ML	34
2.4.1	TensorFlow	35
2.4.2	NumPy	35
2.4.3	Keras	35
2.4.4	Matplotlib	36
2.5	Tecnologias de classificação automatizada de madeira serrada	36
3	MATERIAL E MÉTODOS	42
3.1	Material	43
3.1.1	Conjunto Principal de imagens	43
3.1.2	Conjunto Adicional de imagens	45
3.1.3	Grupos de imagens	46
3.1.3.1	<i>Grupo 1</i>	47
3.1.3.2	<i>Grupo 2</i>	48
3.1.3.3	<i>Grupo 3</i>	49
3.1.4	Equipamento computacional	51
3.1.4.1	<i>Plataforma online CoLAB</i>	51
3.2	Métodos	52
3.2.1	Primeira abordagem	52
3.2.2	Segunda abordagem	56
3.2.3	Terceira abordagem	60
3.2.3.1	<i>Modelos de rede convolucional aplicados</i>	60
3.2.3.1.1	<i>Modelo Adaptado</i>	61
3.2.3.1.2	<i>Modelo Ajustado</i>	62
3.2.3.1.3	<i>Modelos Simplificado</i>	64
3.2.3.1.4	<i>Modelo Final</i>	66
3.2.4	Função de validação com saída formatada	68
4	RESULTADOS E DISCUSSÃO	70
4.1	Primeira abordagem	70
4.2	Segunda abordagem	73
4.2.1	Modelo inicial	73
4.2.2	Modelo final	75

4.3	Terceira abordagem.....	81
4.3.1	Modelo Adaptado	81
4.3.2	Modelo Ajustado	84
4.3.3	Modelo Simplificado	86
4.3.4	Modelo Final	88
5	CONCLUSÕES.....	91
	REFERÊNCIAS	93

1 INTRODUÇÃO

O setor de madeira e seus produtos tem grande força e extrema importância para a economia brasileira. A produção do setor de base florestal, em sua cadeia produtiva industrial e comercial, teve uma participação de aproximadamente 1,4% do Produto Interno Bruto (PIB) nacional em 2015, sendo que a indústria de madeira processada mecanicamente participou da geração de aproximadamente 0,4% do PIB.

Para manter sua representatividade significativa na economia brasileira, o setor de madeira processada mecanicamente procura se modernizar, buscando melhoria dos processos em sua cadeia produtiva e, conseqüentemente, produtos de maior qualidade. Neste aspecto, o setor de madeira serrada - componente do setor de madeira processada mecanicamente - de maneira geral ainda apresenta baixo nível de mecanização e automação em seus processos.

Um dos processos de extrema importância na qualidade final de lâminas, móveis, PMVAs e demais produtos de madeira serrada de Pinus é a classificação da madeira quanto à sua qualidade. Esse é um dos processos que podem ser automatizados durante a produção de madeira serrada ou na seleção das tábuas de madeira para melhor destinação e aproveitamento, antes de sua transformação. Tal classificação se baseia, principalmente, na qualificação e quantificação dos defeitos observados na madeira.

A classificação de tábuas como é feita hoje, por visão humana, além de ser demorada e custosa, apresenta uma baixa porcentagem de acerto. Graduadores humanos que classificam madeira possuem uma taxa de acerto. Processos automatizados procuram aumentar esse nível de acertos, com caracterização mais confiável e, conseqüentemente, ganho no valor agregado das madeiras com alta qualidade.

As alternativas de classificação automatizada, atualmente disponíveis, são muito caras e dificilmente poderiam ser adquiridas por pequenos e médios produtores que utilizam madeira serrada como matéria prima, como o GoldenEye, WoodEye, entre outros. Existem esforços de pesquisa para a construção de sistemas automatizados de classificação. Estudos nacionais mais antigos propõem reconhecer defeitos em imagens de tábuas de Pinus para sua classificação, porém com ressalvas já que os resultados obtidos apresentam limitações de amostras e restrições quanto a aplicação. Trabalhos mais recentes, aplicando processamento de imagens e

aprendizado de máquinas apresentam bons resultados seguindo as classes estabelecidas por normas da ABNT.

A automatização de processos industriais vem ocorrendo desde meados dos anos 70, caracterizando-se como um caminho sem volta. Os avanços e popularização da tecnologia e, conseqüentemente, a facilidade de acesso à mesma, têm motivado e propiciado um uso ainda maior dessa prática. Atualmente a automatização é apoiada pela aplicação de técnicas computacionais e equipamentos computadorizados que proporcionam capacidades tão avançadas às máquinas, que antes eram difíceis de se imaginar.

A rigor, o termo automatização designa uma forma de automação que envolve o uso extensivo de computadores e *softwares* especializados com vista à integração não só das atividades físicas da empresa, como é o caso da automação, mas inclui também o processamento de dados e manipulação da informação, articulando o planejamento de engenharia, concepção de produtos, fluxo de materiais, atividades de marketing e distribuição, entre outros aspectos empresariais e da indústria. A automatização pode ser aplicada a quase todos os processos em todos os ramos de atividade, com ganhos significativos.

As mais recentes soluções tecnológicas desenvolvidas para automatização industrial envolvem, na sua grande maioria, algum tipo de técnica computacional relacionada à Inteligência Artificial (IA). O emprego da inteligência artificial confere uma certa capacidade de tomada de decisão a esses sistemas automatizados, variando o tipo da técnica de acordo com nível de complexidade do processo.

Nesse contexto, a adoção de IA no desenvolvimento de metodologias cresceu rapidamente, devido à produção significativa de soluções para uma ampla gama de aplicações modernas como o reconhecimento de fala, processamento de imagens, visão computacional e computação gráfica.

A aplicação de métodos computacionais está presente em praticamente todos os estudos citados. O processamento de imagens digitais é a principal técnica abordada, pois possibilita analisar as imagens de madeira serrada, procurando identificar e quantificar seus defeitos, como nós, bolsas de resina, rachas, entre outros, e classificar a madeira segundo normas padronizadas.

O aprendizado de máquina (*Machine Learning* - ML) e suas diversas técnicas é um dos recursos computacionais que pode ser utilizado. Nessa linha as Redes Neurais Artificiais - RNA e Máquinas de Vetores Suporte (em inglês *Support Vector*

Machine - SVM) são algumas das técnicas mais difundidas e aplicadas nos sistemas computacionais que necessitam do emprego da inteligência artificial.

Nos últimos anos as técnicas e modelos de aprendizado profundo tem revolucionado diversas áreas da IA, principalmente o aprendizado de máquinas. Também conhecido e mais referenciado como *Deep Learning* - DL, o aprendizado profundo contribuiu, em especial, com a visão computacional. Isso ocorreu por dois motivos principais um deles foi a disponibilização de bases de dados com milhões de imagens; o outro foi a evolução dos computadores, capazes de realizar o processamento dessas imensas bases de dados.

O *Deep Learning* ou aprendizado profundo, é uma técnica relacionada tanto ao aprendizado de máquina quanto a visão computacional, muito estudada em pesquisas recentes, principalmente na análise de imagens digitais, com bons resultados, na maioria dos casos. Aplicações como reconhecimento de padrões, modelagem gráfica, processamento de sinais e otimização são exemplos de áreas de pesquisa que estudam a técnica e os modelos do DL.

O modelo de rede de aprendizado profundo mais estudado e utilizado atualmente é Rede Neural Convolutiva, em inglês *Convolutional Neural Network* - CNN. O processamento de informações visuais, em particular as imagens, é a principal aplicação das CNNs.

Considerando-se o estado da arte atual das redes neurais convolucionais e analisando os resultados apresentados em diversas pesquisas e aplicações que envolvem reconhecimento de padrões e categorização de imagens digitais, apresenta-se a hipótese de que a CNN apresentará resultados melhores do que os obtidos por outras técnicas ou metodologias na classificação de tábuas de madeira serrada de Pinus, quando consideradas todas as regras estabelecidas na NBR 11700, norma que especifica os critérios para classificação de tábuas de madeira serrada de coníferas e define suas classes.

O trabalho teve como principal objetivo classificar automaticamente tábuas de Pinus a partir de suas imagens digitais, aplicando *Deep Learning* e o modelo de redes neurais mais estudado atualmente, a *Convolutional Neural Network* - CNN, tendo como diferencial o atendimento integral à norma da ABNT específica para esse fim.

Ainda, como objetivos específicos, desenvolveu-se modelos CNN para classificação de blocos de imagens com dimensões variadas e panoramas mais

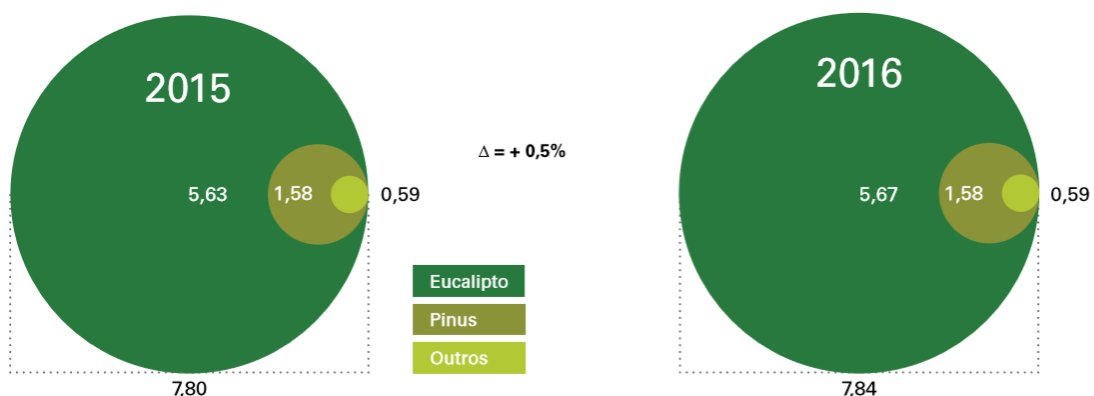
modestos do que o enfrentado no objetivo principal, a fim de experimentar técnicas e arquiteturas intermediárias.

2 REVISÃO DE LITERATURA

2.1 Madeira serrada no Brasil

O setor de florestas plantadas é responsável pela produção de 91% de toda madeira destinada para fins industriais no Brasil e representa 6,2% do PIB industrial, além de ser um dos segmentos com maior potencial de contribuição para a construção da chamada economia verde (IBÁ, 2017). A área de florestas plantadas no Brasil totalizou 7,84 milhões de hectares em 2016, apontando um crescimento de 0,5% em relação ao ano anterior, devido exclusivamente ao aumento das áreas plantadas de eucalipto (Figura 1). Desse total, cerca de 1,6 milhões de hectares são de Pinus, principal matéria prima de madeira processada mecanicamente.

Figura 1 - Área de árvores plantadas em milhões de hectares



Fonte: IBÁ (2017).

A produção de Pinus se concentra na região Sul do país, com área significativa também no Sudeste e plantações menos relevantes em outros estados (Figura 2).

Devido às suas características e exigências climáticas, teve boa adaptação ao planalto das regiões Sul e Sudeste, que possui solo bem drenado e sem deficiência hídrica, o que inclui partes serranas do Rio Grande do Sul, Santa Catarina e Paraná, assim como o sul dos estados de São Paulo e Minas Gerais (SHIMIZU, 2004).

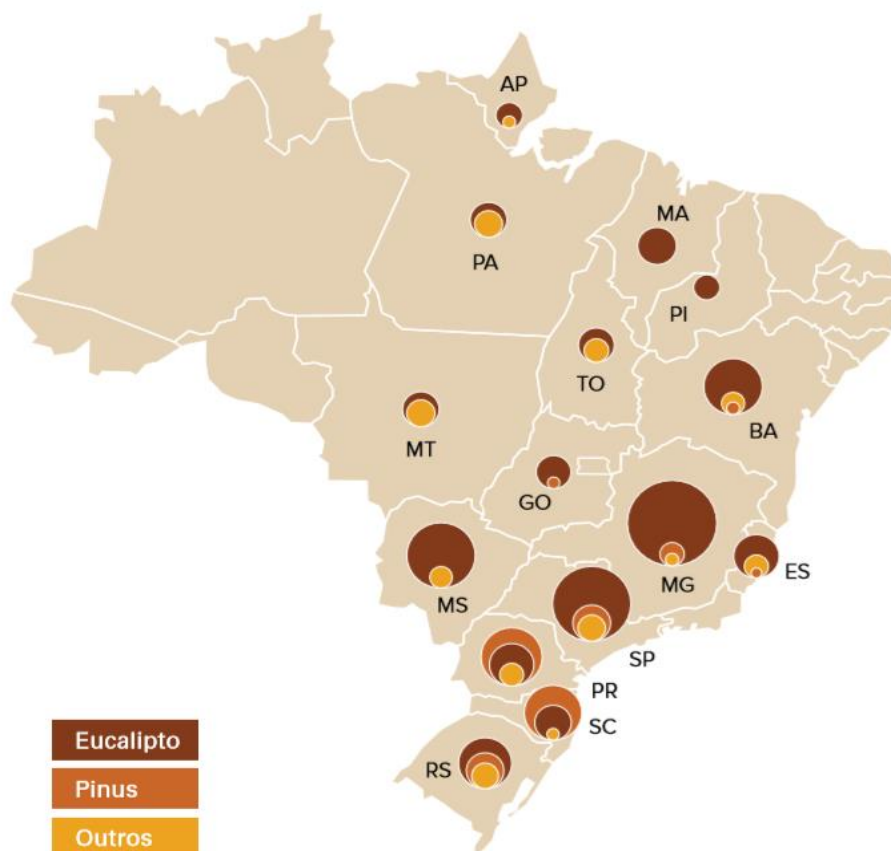
O *Pinus elliottii* é cultivado amplamente em regiões de clima subtropical. Tem múltiplas aplicações na indústria de madeira processada mecanicamente e devido a

seus derivados voláteis breu e terebintina possui outras aplicações comerciais nas indústrias farmacêuticas e químicas (KRONKA; BERTOLANI; PONCE et al., 2005).

Segundo a ABIMCI (2016), o Pinus responde por 76% da matéria prima da indústria de madeira processada mecanicamente para produção de lâminas, compensado, madeira serrada e Produtos de Maior Valor Agregado - PMVA.

A designação madeira serrada é dada aos produtos advindos do desdobro de toras de madeira, como pranchas, sarrafos, ripas, caibros, tábuas, vigas, pontalete, entre outros. Os Produtos de Maior Valor Agregado, são obtidos do reprocessamento da madeira serrada, com o objetivo de agregar valor, gerando produtos como portas, molduras, painel colado lateral e pisos de madeira.

Figura 2 - Áreas de árvores plantadas no Brasil por estado e por gênero, 2016



Fonte: IBÁ (2017).

As dimensões da madeira serrada proveniente do desdobro de coníferas devem seguir as especificações determinadas na norma NBR 12.498, conforme apresentado na Tabela 1.

Tabela 1 – Dimensões padronizadas de madeira serrada de coníferas de reflorestamento

Tipo de peça	Espessura Nominal – e (mm)	Largura Nominal – l (mm)
Caibro	$50 \leq e \leq 100$	$50 \leq l \leq 100$
Pontalete	$e = 75$	$L = 75$
Prancha ou Pranchão	$e \geq 50$	$l > 150$
Pranchinha	$e = 38$	$L \geq 100$
Quadrado	$e = 25$	$L = 25$
Ripa	$e < 25$	$L < 100$
Sarrafo	$25 \leq e < 50$	$25 \leq l < 100$
Tábua	$25 \leq e < 38$	$L \geq 100$

Fonte: NBR 12498 (2017).

A produção de madeira serrada no Brasil em 2017 foi de 8,8 milhões de m³. Desse total, 74% foi destinado ao mercado interno (IBÁ, 2017).

2.2 Defeitos e classificação de tábuas de coníferas

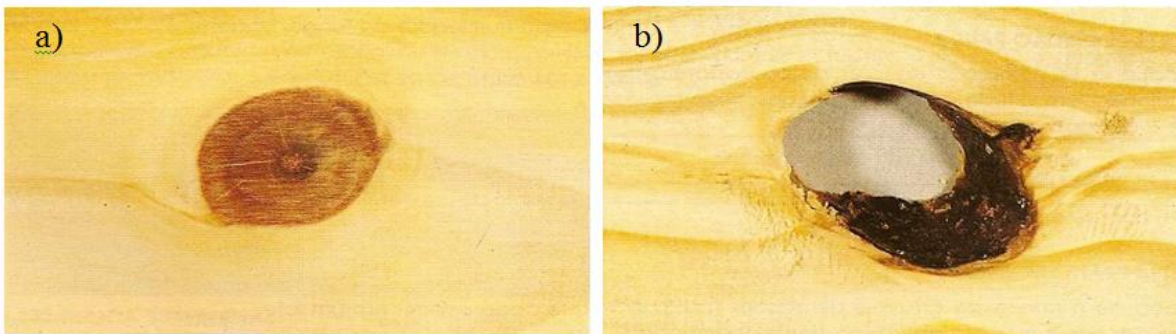
A madeira serrada de Pinus apresenta defeitos naturais da própria constituição morfológica da madeira, além de outros gerados por agentes externos. Os defeitos naturais mais comumente encontrados, são os nós; dos defeitos relacionados a agentes externos são os mais comuns o furo de inseto e os empenamentos e fendilhamentos.

A classificação de tábuas de madeira serrada é feita através da medição e quantificação dos defeitos apresentados nas duas faces da amostra considerando-se, para efeitos finais, a face mais prejudicada. A incidência desses defeitos, tanto em quantidade quanto em suas dimensões, determina a classe da madeira segundo normas estabelecidas.

A Associação Brasileira de Normas Técnicas (ABNT) disponibiliza normas que preconizam as práticas para a quantificação e classificação de madeira serrada de coníferas no Brasil. A norma NBR-12297 (ABNT, 1991b) elenca os seguintes defeitos: nós, bolsas de resina, empenamentos, esmoado, furos de insetos inativos, rachas, medição e quantificação da medula, desbitolamento e fendilhado, medição e quantificação da grã e fixa as condições exigíveis para suas medições em madeira serrada de coníferas provenientes de reflorestamento, para uso geral.

Por ser o defeito mais recorrente nas tábuas, o nó se torna o principal ponto de interesse na classificação de madeiras de coníferas. Além de ter grande incidência, o nó possui diversos tipos variando quanto a sua forma, coloração, posição na peça e estágio. A norma os classifica como: nó de face, nó de quina, nó de gravata, grupo de nós, feixe de nós, nó firme, nó cariado, nó solto e nó vazado. As Figuras 3a e 3b ilustram o defeito caracterizado como nó firme e nó solto, respectivamente, enquanto a Figura 4 ilustra um nó cariado em tábuas de madeira serradas de coníferas (ABNT, 1991b).

Figura 3 – a) Defeito caracterizado como nó firme; b) Defeito caracterizado como nó vazado



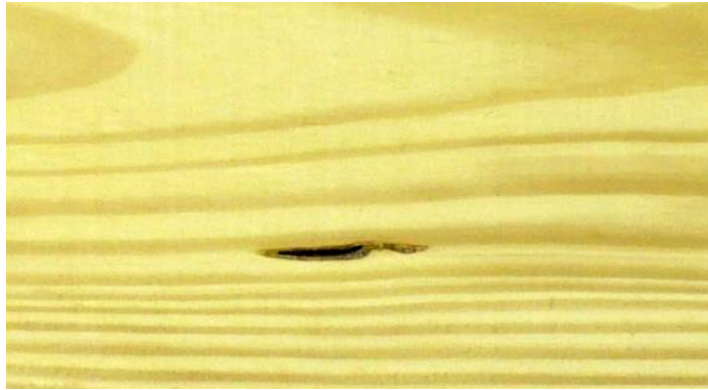
Fonte: ABPM (2001).

Figura 4 - Defeito caracterizado como nó cariado



As Figuras 5 e 6 ilustram os defeitos caracterizados como bolsa de resina e medula em tábuas de madeira serradas de coníferas. Ilustrações dos demais defeitos podem ser encontradas na norma NBR-12297 e catálogos da (ABPM, 2001).

Figura 5 - Defeito caracterizado como bolsa de resina.



Fonte: ABPM (2001)

Figura 6 - Defeito caracterizado como medula.



As Figuras 7 e 8 apresentam imagens de tábuas classificadas como Super e Terceira classe. Nesses casos ilustrados as madeiras apresentam o mesmo defeito, o nó, porém o tipo do nó (firme ou solto), determina o enquadramento da tábua em classes distintas, podendo variar da classe Super, classe mais elevada, para terceira classe, que representa a pior classe estabelecida pela norma.

Figura 7 – Presença de nó firme em tábua de classe Super

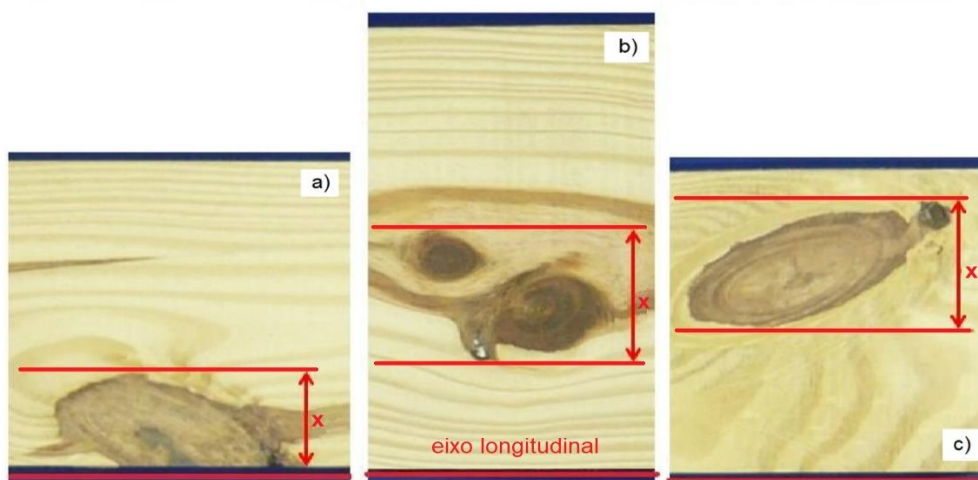


Figura 8 - Presença de nó vazado em tábuas de terceira classe



Na avaliação de um nó ou de um conjunto de nós deve-se, inicialmente, medir o seu diâmetro e, em seguida, calcular a porcentagem que esse nó ocupa em uma determinada porção da madeira. O diâmetro de um nó é a distância entre duas linhas paralelas entre si, na direção longitudinal da peça serrada (eixo longitudinal), compreendendo o nó ou grupo de nós (Figura 9). A Figura 9a apresenta a medição de um nó de quina, a 9b de um grupo de nós e a 9c de um nó de face. Também podem ser observadas as linhas dos eixos longitudinais e paralelas, além da distância representada pela letra X.

Figura 9 – Exemplos de medição de nós ou grupo de nós



Fonte: Adaptado de ALMEIDA (2014)

A NBR 11700 (ABNT, 1991a) define as classes e estabelece procedimentos e forma de medição dos defeitos, também determina os limites para enquadramento em cada uma das classes de qualidade da madeira serrada de coníferas provenientes de reflorestamento. As classes definidas pela norma são: super, extra, primeira classe, segunda classe e terceira classe. A Tabela 2 apresenta uma síntese dos defeitos e das cinco classes de madeira serrada estabelecidas pelo código normativo.

Tabela 2 – Classificação de madeira serrada de coníferas provenientes de reflorestamento

Defeito	Característica	Classes de qualidade			Terceira	
		Qualidade Super	Qualidade Extra	Primeira		Segunda
nós	Firme, feixe de nós, grupos de nós	Soma dos diâmetros	$\leq 50\% l_1/m$	$\leq 70\% l_1/m$	$\leq 90\% l_1/m$	Sem restrições
	Soltos, Cariados de gravata	Soma dos diâmetros	Não são permitidos	Não são permitidos	$\leq 30\% l_1/m$	$\leq 200\% l_1/m$
	Vazados	Soma das larguras	Não são permitidos	Não são permitidos	$\leq 10\% l_1/m$	$\leq 30\% l_1/m$
Bolsa de resinas	Soma comprimentos	Não é permitida	$\leq 5\% l_1$	$\leq 10\% l_1$	$\leq 10\% l_1$	$\leq 15\% l_1$
	Larguras	Não é permitida	$\leq 25\% L_1$	Sem restrições	Sem restrições	
Medula	Soma comprimentos	Não é permitida	$\leq 12\text{ mm}$	Permitido sem restrições	Permitido sem restrições	
	Soma comprimentos	Não são permitidos	$\leq 15\% l_1$			
Rachas	Soma comprimentos	Não são permitidos	$\leq 10\% L_1$	$\leq 20\% L_1$	$\leq 20\% L_1$	$\leq 50\% L_1$
	Comprimen. individual		0,15 m	0,20 m	0,20 m	0,50 m
Fendilhado	Profundidade	Não afere aplainamento		Permitido sem restrições	Permitido sem restrições	
	Espessura	$\leq 15\% e_1$	$\leq 15\% e_1$	$\leq 30\% e_1$	$\leq 30\% e_1$	
Esmoado	Largura	$\leq 3\% l_1$	$\leq 5\% l_1$	$\leq 10\% l_1$	$\leq 10\% l_1$	$\leq 20\% l_1$
	Soma comprimentos	$\leq 20\% L_1$	$\leq 20\% L_1$	$\leq 30\% L_1$	$\leq 30\% L_1$	$\leq 40\% L_1$
	Número de quinas	1	1	1	1	2
Mancha marron	Área afetada	Não são permitidos	$\leq 10\% A_1$	Permitido sem restrições	Permitido sem restrições	
			$\leq 10\% A_1$	$\leq 25\% A_1$	$\leq 50\% A_1$	
			Não é permitida	$\leq 10\% A_1$	$\leq 15\% A_1$	
Furos insetos inativos	Presença	Não são permitidos	$\leq 15\% A_1$	$\leq 30\% A_1$	$\leq 30\% A_1$	$\leq 50\% A_1$
			Não são permitidos			
Galeria, furos insetos ativos, podridão, torc.	Presença		Permitidos sem restrições			
Esporas, exsudação, grã entrecruzada e inclinação	Quantidade	Sem restrições	≤ 6	≤ 7	≤ 7	Sem restrições
			Permitidos sem restrições			

Legendas e abreviações: L_1 : comprimento real da peça; l_1 : largura real da peça; e_1 : espessura real da peça; A_1 : área da face classificada da peça; comprimen.: comprimento; torc.: torcimento.

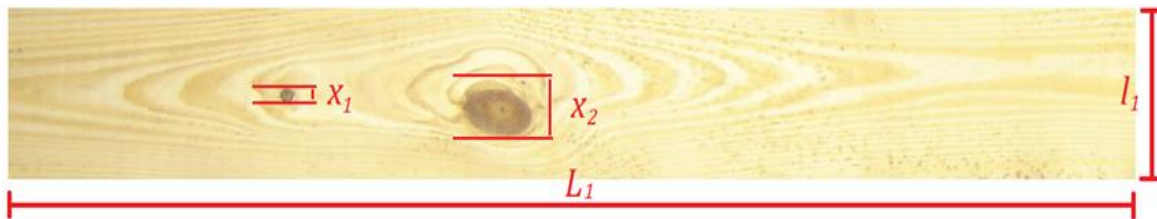
Fonte: Associação Brasileira de Normas Técnicas, NBR 11700 (1991a).

A porcentagem de nós (N_s) em uma tábua é avaliado com a Equação 1 (ABNT, 1991a).

$$N_s(\%) = \frac{(x_1 + x_2 + \dots + x_i)}{L_1 + l_1} * 100 \quad (1)$$

onde x_1 é o i -ésimo diâmetro x de nó calculado na tábua, L_1 é o comprimento da peça de madeira e l_1 é a largura da peça (Figura 10).

Figura 10 – Exemplos de medição de nós



2.3 Aprendizado de máquina

O aprendizado de máquina, ou *Machine Learning* – ML, é uma área da Inteligência Artificial - IA dedicada ao estudo de algoritmos que buscam identificar padrões em conjuntos de dados, produzindo um modelo adaptado que pode ser utilizado em um novo conjunto de informações. Por essas características, muitas são as aplicações das técnicas de inteligência artificial por meio do aprendizado de máquina. Essas técnicas são aplicadas há algum tempo em várias áreas como mineração de dados, classificação, projeto, controle, modelagem, otimização, diagnóstico e previsão de máquinas (KRIZHEVSKY; SUTZKEVER; HINTON, 2012).

Basicamente o aprendizado de máquina busca padrões para classificar conjuntos de dados segundo esses padrões aprendidos. São diversas as técnicas que possibilitam, a partir do aprendizado, classificar um determinado conjunto de dados. Entre elas, pode-se destacar as técnicas baseadas no aprendizado de máquina, como Redes Neurais Artificiais (RNAs), Algoritmos Genéticos (AGs), Árvores de Decisão, Raciocínio Baseado em Casos (RBC) e máquinas de vetores suporte (MONARD; BARANAUSKAS, 2003).

Segundo Lorena (2006), as técnicas de aprendizado de máquina aplicam o princípio da indução. Essa técnica de inferência permite obter conclusões genéricas, de um determinado conjunto de informações fornecido como exemplo. Assim é

possível, a partir desses dados de exemplo, deduzir informações sobre o contexto desses dados. Há dois tipos principais de aprendizado indutivo, o supervisionado e o não-supervisionado. No modo de aprendizado supervisionado cada classe do conjunto de exemplo é conhecida, ou seja, os dados de entrada são previamente rotulados. Esse método passa a ideia da existência de um tutor durante a aprendizagem. As informações são fornecidas em pares, contendo cada par o dado de entrada e acompanhado da saída desejada. Espera-se com isso que o modelo encontrado durante o aprendizado seja capaz de gerar saídas corretas para exemplos não apresentados (HAYKIN, 2001).

No método de aprendizado de máquina não-supervisionado, os dados são inseridos no sistema sem um rótulo, ou seja, a classe a que pertence o conjunto de exemplos não é conhecida. Nesse caso o próprio sistema deve decidir e criar classes ou agrupar algumas já existentes. O agrupamento dos exemplos em determinadas classes é realizado a partir de uma métrica de similaridade. Essa técnica de aprendizado é utilizada principalmente quando se deseja encontrar padrões ou tendências em conjuntos de dados nos quais não se consegue criar uma associação entre os mesmos de outra forma, ou quando, devido a quantidade de dados, seria inviável realizar tal procedimento sem o auxílio de métodos computacionais. Nesse caso o objetivo do método é auxiliar no entendimento dos próprios dados (LORENA, 2006).

As técnicas de aprendizado de máquina apresentam algumas características em comum para que possam funcionar, ou seja, precisa-se de alguns componentes básicos para que se tenha um método de aprendizado de máquina. Esses componentes são, basicamente, os dados de entrada, um algoritmo desenvolvido para buscar padrões nesse conjunto de dados, um vetor de características que descreva os dados, função que manipula os valores envolvidos e, por fim, dados de saída. O vetor de características é formado por valores que representam aspectos dos exemplos, sendo que cada atributo desse vetor expressa um determinado aspecto. Um atributo pode ser nominal, podendo assumir valores fora de ordem, ou contínuo, que assume valores em uma sequência, como os números reais. Ao ser formado um conjunto de dados de um domínio, deve-se subdividi-lo em dois grupos, um para o treinamento, que será utilizado para o aprendizado do conceito e outro para os testes, que será aplicado após o treinamento, para testar a efetividade do modelo de

aprendizado. Assim se pode determinar as porcentagens de acerto e erro da técnica de aprendizado de máquina implementada (MONARD; BARANAUSKAS, 2003).

Alguns aspectos importantes devem ser observados durante o processo de criação de um modelo de aprendizado de máquinas, para que o modelo se mostre eficiente. Os conjuntos de dados, em sua maioria, apresentam erros ou ruídos. Um bom modelo de ML deve ser capaz de lidar com esses erros sem que isso interfira em sua eficácia. A técnica deve ainda minimizar o impacto de *outliers*, que são dados que fogem, em muito, dos demais do conjunto. Tais exceções podem ser imperfeições dos dados, ou casos muito raros do domínio (LORENA, 2006).

A possibilidade de superajustamento (*overfitting*) e subajustamento (*underfitting*) são outros aspectos relevantes a serem considerados, ao se modelar uma técnica de aprendizado. O *overfitting* ocorre quando um modelo atinge altas taxas de acerto no conjunto de treinamento e baixas taxas no conjunto de testes ou novos dados. O *underfitting* é configurado quando o modelo passa a apresentar baixo índice de acerto mesmo nos dados de treinamento. Prever e trabalhar esses aspectos conferem uma capacidade importante à técnica, a de generalização do modelo, que consiste na capacidade de classificar novos dados do mesmo domínio, sem que esses tenham passado por treinamento. Esses fenômenos podem ocorrer quando o modelo obtido é muito simples ou quando o conjunto de dados de exemplo disponibilizado para treinamento não representa todas as variações que comumente ocorrem no domínio (MONARD; BARANAUSKAS, 2003).

2.3.1 Rede Neural Artificial - RNA

As redes neurais artificiais são baseadas na estrutura das redes neurais biológicas. Seu funcionamento é inspirado no sistema nervoso biológico. O cérebro humano é formado por um número substancial de componentes interligados, chamados neurônios. Esse órgão do corpo humano reúne e processa as informações, adaptando o arranjo de suas conexões de acordo com os problemas a ele apresentados. A capacidade de solução de problemas de um ser inteligente é afetada, em parte, pela quantidade e qualidade das ligações entre esses neurônios. De forma análoga, a capacidade de uma RNA é dada pelas associações entre esses componentes artificiais, também chamados de neurônios, que compõem a rede neural

artificial. Uma rede neural artificial pode ser preparada para executar uma ação específica, modificando-se as estimativas das conexões, ou pesos, entre os componentes. Normalmente, as RNAs são adaptadas ou treinadas, para que uma entrada específica ative uma determinada saída esperada. Comumente, inúmeros conjuntos de entrada e saída são utilizados como parte dessa aprendizagem supervisionada para treinar uma rede neural (KANTA; SANGWAN, 2015).

O ponto de partida para os estudos sobre os neurônios artificiais, e conseqüentemente para as RNAs, foi o artigo de McCulloch e Pitts (1943), onde os autores afirmaram que, com um número suficiente de neurônios artificiais é possível representar expressões lógicas. Em seguida, outras obras acrescentam contribuições no desenvolvimento da teoria das redes neurais artificiais, como os trabalhos de Hebb (1949) e Rosenblatt (1958). A Figura 11 apresenta a proposta original para um neurônio artificial, já a Figura 12 traz uma representação mais completa e atual.

Figura 11 – Representação do neurônio artificial McCulloch e Pitts em 1943

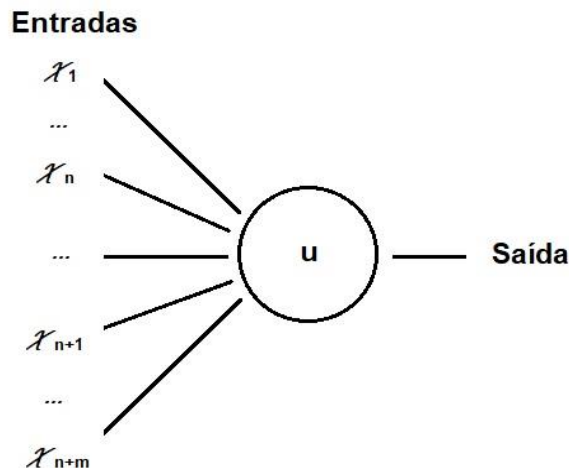
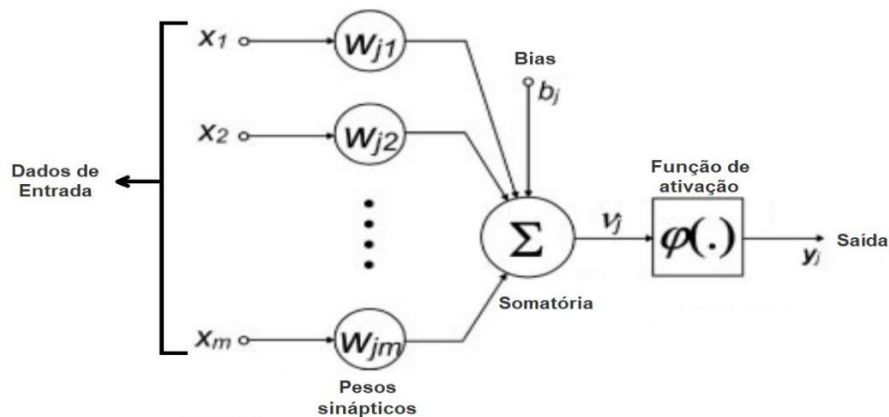


Figura 12 – Modelo atual de neurônio artificial



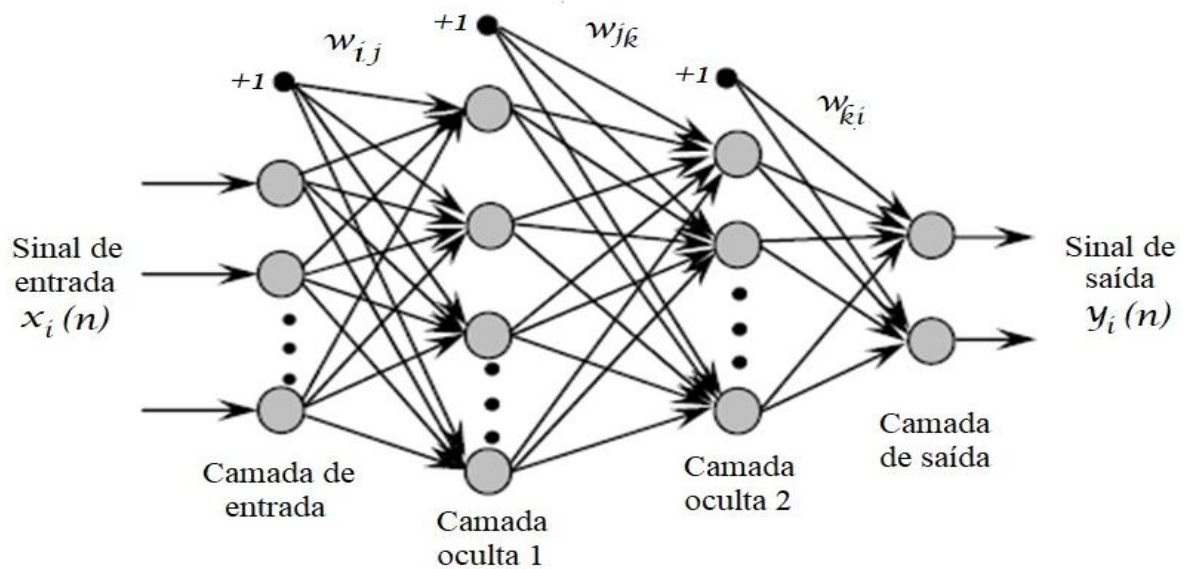
Fonte: Adaptado de HAYKIN (2001).

Um dos primeiros modelos de redes neurais artificiais foi proposto por Rosenblatt (1958), recebendo o nome de *Perceptron*. Sua arquitetura também é baseada no neurônio biológico, como o neurônio proposto por McCulloch (1943). Os modelos de RNAs podem ser construídos com diversas arquiteturas, as quais dependem do tipo de problema no qual a rede será aplicada. Entre vários parâmetros que definem uma arquitetura destacam-se o número de camadas, número de nós em cada camada, o tipo de conexão entre os nós. O objetivo dessa rede é classificar as entradas (ou estímulos) em duas classes em um hiperplano. O processo de aprendizagem, para esse tipo de modelo de rede, pode ser a supervisionada ou a não supervisionada (KOVÁCS, 1996).

Uma variação do *Perceptron* é o *Multilayer Perceptron* (MLP), que é um dos modelos de RNAs mais utilizados. Esse modelo aplica o algoritmo de treinamento baseado na retropropagação de erro. O funcionamento desse algoritmo consiste em, tendo padrão apresentado à camada de entrada da rede, ele é processado camada por camada até a camada de saída fornecer a resposta processada. Caso essa resposta seja diferente do esperado, o algoritmo devolve o erro para que seja efetuado um ajuste nos valores dos pesos, e assim inicia-se o processo novamente. O algoritmo efetua o cálculo dos erros, nas camadas intermediárias, permitindo o ajuste dos pesos em proporção aos valores das funções de ativação das conexões entre as camadas (HAYKIN, 2001).

Um modelo de redes MLP é formado basicamente por uma função de ativação não linear (sigmoideal), uma ou mais camadas ocultas de neurônios, e uma camada de conexão todos com todos. Aplicações de RNAs são variadas e não se restringem a um campo em particular por oferecerem recursos como: não linearidade, adaptabilidade, mapeamento de entrada-saída, resposta a evidências, informações contextuais e tolerância a falhas (MITCHELL, 1997). O modelo de redes de multicamadas *Perceptron* é representado na Figura 13.

Figura 13 – Modelo de neurônio artificial MLP



Fonte: Adaptado de NIMAJE; TRIPHATY (2015).

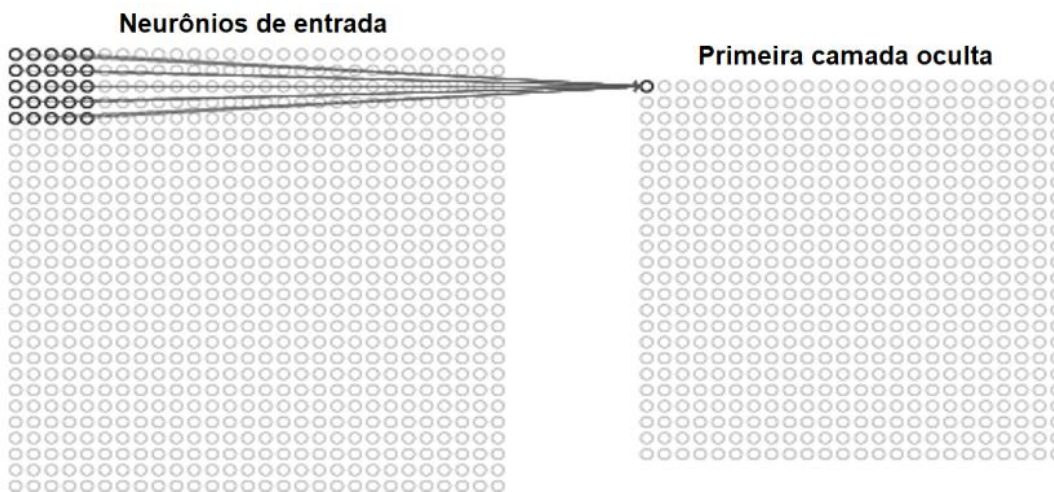
2.3.2 *Deep Learning* e Redes Neurais Convolucionais

Uma nova abordagem relacionada com aprendizado de máquina e redes neurais, chamada *Deep Learning* (DL) ou Aprendizado Profundo, tem ganhado muita força. O *Deep Learning* pode ser entendido como uma classe de técnicas de aprendizado de máquina que explora muitas camadas de informação não lineares. Basicamente, o *DL* 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 sucessivamente mais alta de camadas mais abstratas, sendo um modelo que consiste de múltiplas camadas de processamento de informação não linear e também um método de aprendizado supervisionado e não supervisionado (DENG; YU, 2014).

Devido a suas características, a implementação e treinamento de uma rede neural tradicional, utilizando *deep learning*, pode ser bastante complexo e demorado, dependendo da quantidade e características dos dados de entrada, o que na maioria dos casos é extensa e complexa. Como alternativa utiliza-se um modelo de redes neurais de aprendizado profundo chamado de Rede Neural Convolutacional artificial, ou

simplesmente rede convolucional, do inglês *Convolutional Neural Network* (CNN). As redes convolucionais têm uma diferença fundamental em relação as redes neurais comuns, onde os neurônios utilizam uma ligação do tipo “todos-todos” de uma camada para outra. Nessa técnica, um recurso chamado de campos receptivos locais, ao invés de ligar cada um dos neurônios da primeira camada com todos os neurônios da segunda camada, faz a ligação de um conjunto de neurônios da primeira camada (matriz de convolução) com um neurônio da segunda camada, conforme ilustrado na Figura 14 (PONTI; COSTA, 2017).

Figura 14 - Esquema de rede neural convolucional



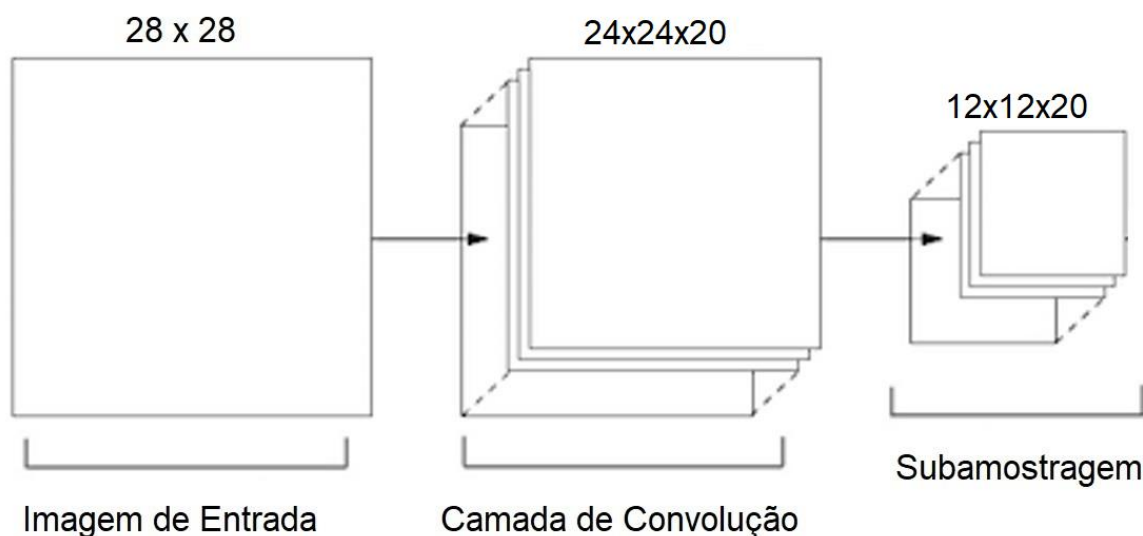
Usando como exemplo uma imagem de 28x28 pixels, a técnica consistiria em selecionar um determinado campo dessa imagem com um tamanho específico, por exemplo um campo de 5x5, onde esse campo teria seus valores filtrados para um único neurônio da próxima camada. Esse processo de filtragem é chamado de convolução. No momento em que essa imagem é convolucionada são gerados mapas de característica, ou *feature maps*. No caso da entrada de 28x28 pixels, tendo um campo de 5x5 e um passo de 1 neurônio, resultariam mapas de características na camada de convolução com o tamanho de 24x24 neurônios. Todos os neurônios de um determinado mapa de características receberão um mesmo peso e valor de passagem, esse é outro conceito importante das redes convolucionais e é chamado de pesos compartilhados. Ele é quem permite que cada mapa corresponda a uma característica específica (NIELSEN, 2015). Neste exemplo, percebe-se que a dimensão 28x28 da imagem na camada de entrada cai para 24x24 na segunda camada, isso é uma característica da convolução, essa redução pode ser evitada, se necessário, com algumas técnicas.

2.3.2.2 Subamostragem (pooling)

Outro recurso importante é subamostragem, ilustrado na Figura 16. No modelo de rede convolucional, a camada seguinte a da convolução é camada de subamostragem. O objetivo dessa camada 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 subamostragem, geralmente usa-se um campo de 2x2 desses mapas e o valor dos campos são generalizados para um único neurônio dessa camada (NIELSEN, 2018).

Essa generalização pode ser feita de várias formas, sendo esse processo mais conhecido como *pooling*. O método mais utilizado é o *maxpooling*, que consiste em utilizar o valor mais significativo do campo 2x2 e atribuir ao neurônio da camada de subamostragem ou agrupamento (*pooling*). Diferentemente da convolução, os campos da subamostragem 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 subamostragem para o modelo sugerido teria 12x12 neurônios, levando em conta que a camada de convolução apresentou 24x24 pixels (NIELSEN, 2018).

Figura 16 - Camadas de convolução e subamostragem



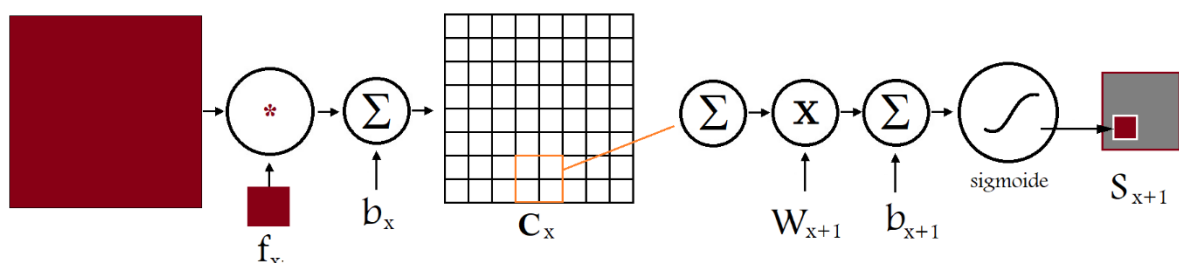
Fonte: Adaptado de NIELSEN (2018).

2.3.2.3 Função de ativação por retificação linear

Segundo Arel et al. (2010), que explanam sobre os conceitos aplicados na CNN, a convolução consiste em convolver dados de uma imagem de entrada do primeiro estágio ou mapa de características para estágios posteriores, aplicando um filtro (*kernel* ou núcleo) treinável f_x , adicionando então um viés b_x , também treinável, para produzir a camada C_x de convolução. Já a subamostragem (*subsampling*) consiste em somar uma vizinhança (de quatro pixels por exemplo) ponderado por escalar W_{x+1} , adicionando o viés treinável b_{x+1} e passando por uma função de ativação sigmoide para produzir um mapa de características aproximadamente duas vezes menos que S_{x+1} (Figura 17).

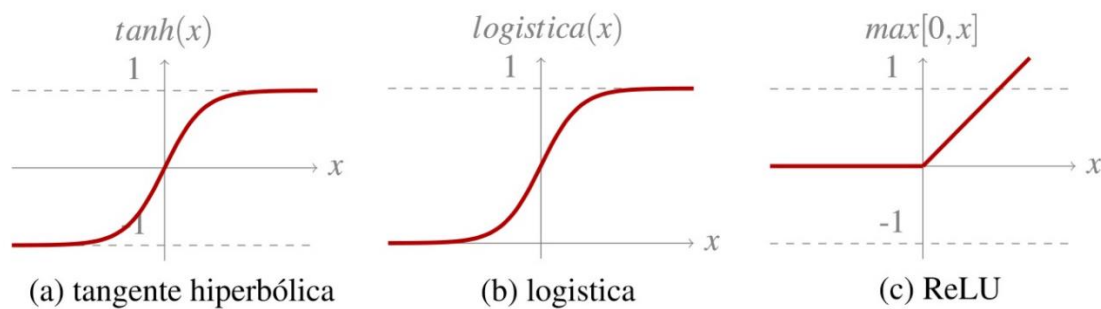
De acordo com Ponti e Costa (2017), a função de ativação mais indicada para as camadas ocultas em modelos de *deep learning* é a função retificadora linear (*rectified linear function*, ReLU), pois facilita o processo de treinamento. Essa função é escolhida como opção às funções sigmoideais, como a tangente hiperbólica e a logística, dado que essas funções saturam a partir de um determinado ponto. Em contrapartida, a função ReLU age sobre os valores negativos convertendo-os em zero, preservando os valores positivos. Na Figura 18 é ilustrado o comportamento das funções de ativação citadas, constatando o cancelamento dos valores negativos da função ReLU, seguindo linearmente para os valores positivos.

Figura 17 - Parâmetros no processo de convolução



Fonte: Adaptado de AREL et al., (2010).

Figura 18 - Comportamento das funções de ativação



Fonte: Adaptado de PONTI; COSTA (2017)

2.3.2.4 Regularização L2

A regularização L2 é uma das técnicas usadas para se evitar ou minimizar o impacto do *overfitting*, recomendada principalmente para conjuntos de dados limitado ou quando não podemos aumentar significativamente o número de camadas do modelo de rede. Sabe-se que, utilizando redes com muitas camadas ou dispondo de um conjunto grande de dados, a chance de se obter sucesso com o modelo aumenta. A regularização L2 consiste em adicionar à função de custo uma regularização aos pesos (por meio da soma dos quadrados dos pesos), evitando poucos valores altos nos parâmetros (PONTI; COSTA, 2017).

2.3.2.5 Métricas de qualidade do modelo

A avaliação da qualidade de modelos de Redes Neurais Convolucionais, quanto a sua capacidade de produzir bons resultados de predição, é feita comumente a partir do cálculo da acurácia e do custo. Tais cálculos resultam um valor numérico que expressa a qualidade de predição de um modelo. O cálculo da acurácia é feito dividindo-se a soma das predições positivas corretas com as negativas corretas pelo total de amostras avaliadas. O resultado é um número entre 0 e 1, quanto mais próximo de 1 melhor é o modelo (PONTI; COSTA, 2017).

O *loss* tem seu cálculo efetuado pela função custo, conhecida como *loss function* ou *cost function*, em inglês. É chamado de custo por seu resultado representa o custo assumido ao se aceitar um determinado modelo para efetuar uma classificação. Em outras palavras, denota a distância entre o resultado obtido e a classificação ideal. A

função atua sobre exemplos rotulados e se ajusta conforme o treinamento evolui, diminuindo gradativamente seu valor, por esse motivo é considerada função empírica de custo (PONTI; COSTA, 2017).

São várias as funções empregadas com essa finalidade, sendo a de entropia cruzada, *cross-entropy* em inglês, a mais utilizada. A função, tomando-se uma única imagem cuja classificação correta é representada por y , tem predição y' representada por $f(x) = y'_j$, calcula a soma das entropias cruzadas entre a classe real e a classe da predição, das j classes, de acordo com a Equação 4:

$$l^{(ce)} = - \sum_j y_j * \log(y'_j + \varepsilon) \quad (4)$$

ce cross-entropy, $\varepsilon \ll 1$ é uma variável de segurança para evitar $\log(0)$, assumindo-se comumente $\varepsilon = 0,0001$. Quanto menor o valor de $l^{(ce)}$ menor é o custo e, por consequência, melhor é o modelo.

2.3.2.6 Algoritmos de otimização

Outro processo necessário para o bom desempenho de modelos de redes neurais convolucionais é o uso de um algoritmo de otimização, que consiste basicamente em funções que atuam sobre os valores dos pesos e viés no intuito de minimizar o erro, que é calculado pela função custo (*loss_function*). O valor retornado pela função de perda é usado na retropropagação do erro, para ajustar os valores dos parâmetros treináveis definidos aleatoriamente no início da execução do modelo. Devido a esse fator de aleatoriedade, deve-se buscar um algoritmo que ajuste os valores de modo que faça o modelo convergir, objetivando proporcionar boas predições.

Um dos algoritmos de otimização mais adotados é o Adam, cujo nome é derivado do conceito *Adaptive Moment Estimation*. Esse algoritmo executa a descida do gradiente, evitando o mínimo local, dando mais importância aos parâmetros menos utilizados. Assim altera o valor da largura do passo com base nesses valores, ou seja, os valores menos utilizados são os que mais impactarão no ajuste do valor desse passo, em busca do mínimo global. Aplicar o algoritmo Adam inclui definir a *learning*

rate (LR), que diminui o tamanho do ajuste nos parâmetros para evitar grandes saltos (*overshooting*). A função de atualização dos parâmetros treináveis usa a derivada da função de perda (*loss_function*), essa operação pode resultar em valores muito altos, por esse motivo o algoritmo de otimização subtrai a *learning rate* (LR) dessa derivada, minimizando o risco de encontrar um mínimo local (KINGMA; LEI, 2015).

2.4 Ferramentas e bibliotecas aplicadas a ML

Os termos ferramenta e biblioteca são comuns em computação e tem significados específicos na área. Uma ferramenta é um programa de computador construído especificamente para auxiliar no desenvolvimento e manutenção de outros *softwares* e aplicativos. A maioria dessas ferramentas podem ser complementadas com funções adicionais, integrando-as ao próprio código do programa. Existem também conjuntos de funções prontos e disponíveis nos repositórios na *web*, desenvolvidos para resolver problemas recorrentes, que podem ser agregados a ferramenta para aumentar sua capacidade e seu potencial de aplicação. Esses conjuntos de funções, que geralmente são desenvolvidos e disponibilizados por colaboradores ou organizações, são chamados de bibliotecas. São inúmeras as ferramentas e bibliotecas disponíveis gratuitamente na *web*, que facilitam o estudo e a criação de modelos de aprendizado de máquina e redes neurais. Muitas bibliotecas são desenvolvidas para serem integradas à linguagem de programação Python, que é a linguagem mais indicada e amplamente usada em aplicações de aprendizado de máquina (WALT; COLBERT; VAROQUAUX, 2011).

É comum uma ferramenta ter necessidade de componentes adicionais para aplicações específicas, portanto é muito importante atentar-se a estas dependências, pois disso depende, na maioria das vezes, a eficiência da ferramenta. Para a aplicação da linguagem Python no desenvolvimento de modelos de redes neurais e aprendizado de máquina é necessária a instalação de alguns complementos que integrarão a linguagem, aumentando seus recursos de forma a disponibilizar bibliotecas de funções específicas para esse fim. As bibliotecas Keras, TensorFlow e NumPy são algumas das opções disponíveis mais aplicadas nesse contexto (ABADI et al., 2016).

2.4.1 TensorFlow

TensorFlow é uma ferramenta de interface de desenvolvimento que tem como objetivo a elaboração e implementação de algoritmos de aprendizado de máquina. É um sistema bastante flexível e que pode ser usado para expressar vários tipos de algoritmos, incluindo algoritmos de *deep learning*. Esta ferramenta 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. É uma ferramenta em constante desenvolvimento e de código aberto (ABADI et al., 2016). Essa ferramenta também realiza a computação numérica de mapas de fluxo de dados trabalhando com matrizes multidimensionais dinâmicas que interagem com os nós do mapa, que em geral implementam modelos de operações matemáticas (TENSORFLOW, 2016).

2.4.2 NumPy

A linguagem de programação Python, em sua configuração original como é disponibilizada, possui vários recursos deixando-a bastante poderosa ao que se destina, porém ela não é ideal para computação numérica de alta performance. Por esse motivo, foi desenvolvido uma estrutura de dados para a computação eficiente de matrizes (ou *arrays*) numéricas destinada a linguagem Python. A sua distribuição é na forma de biblioteca e gratuita. Essa biblioteca recebeu o nome de NumPy. A biblioteca abrange tanto a estrutura de matrizes, muito utilizada em reconhecimento de padrões em imagens, quanto um conjunto de funções matemáticas para manipulação dessas matrizes ou *arrays*. 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).

2.4.3 Keras

A Keras é uma biblioteca de código aberto para aplicação de redes neurais, desenvolvida em linguagem Python, possui um conjunto de funções de alto nível

podendo trabalhar tanto com recursos do Theano quanto do TensorFlow. O Theano é uma biblioteca com recursos parecidos com os do Tensorflow, porém não é muito utilizada nos estudos mais recentes, devido a algumas limitações superadas pelo seu sucessor, o Tensorflow (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).

2.4.4 Matplotlib

É uma biblioteca 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. Também baseada em Python, é bastante aplicada em trabalhos que envolvem aprendizado de máquina. 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. Possui basicamente a função de plotar gráficos a partir de dados gerados por redes neurais artificiais. A ferramenta possibilita gerar, entre outros tipos de representações gráficas, histogramas, planos, espectros de força, gráficos de barra e gráficos de erro (HUNTER, 2007).

2.5 Tecnologias de classificação automatizada de madeira serrada

A maioria das tecnologias desenvolvidas para classificação automatizada de madeira usam aprendizado de máquina. Dentre as técnicas destacam-se as Redes Neurais, também chamadas de Redes Neurais Artificiais - RNA (em inglês *Artificial Neural Networks* - ANN), o Aprendizado Profundo (*Deep Learning* - DL) e as Redes Neurais Convolucionais (*Convolutional Neural Network* - CNN). Outras técnicas mais tradicionais e comumente usadas são K-Vizinhos Próximos (*K-Nearest Neighbor* – KNN), Máquina de Vetores Suporte (SVM) e Árvores de Decisão (*Decision Trees* – DT). Todas essas técnicas estão relacionadas a um campo de estudos mais

abrangente, a Inteligência Artificial (MCCULLOCH; PITTS, 1943; KOVÁCS, 1996; HAYKIN, 2001; MONARD; BARANAUSKAS, 2003; DENG; YU, 2014; OBULESU; MAHENDRA; REDDY, 2018).

A classificação de tábuas de madeira a partir de suas imagens envolve uma série de desafios. O primeiro é a identificação e classificação dos defeitos presentes na tábua. Outros desafios são a definição do padrão referencial a ser adotado para classificação e a disponibilidade de uma base de imagens em quantidade suficiente para ser aplicada nas fases de treinamento e testes. Por esses motivos poucos trabalhos propõem essa solução, considerando todas as variantes reais e normatizações que envolvem a área.

Muitos trabalhos relevantes com aprendizado de máquinas para classificação de madeira a partir de imagens foram desenvolvidos desde os anos de 1980. Koivo et al. (1989) desenvolveram um algoritmo para classificar nove classes e oito tipos de defeitos em placas de carvalho vermelho, aplicou árvore hierárquica atingindo 97,2% de acerto na classificação. Outro trabalho pioneiro usando aprendizado de máquina e um método de visão computacional foi desenvolvido por Kauppinen (1999). Por um bom tempo esse trabalho foi o único a propor criação de um classificador por meio de aprendizado de máquina. O trabalho alcançou uma taxa de acerto de 80% na classificação dos defeitos e de 71% nas classificações das tábuas de madeira. As imagens de madeira foram subdivididas em blocos de 32x32 pixels, para a detecção e classificação dos defeitos. A técnica se baseou em extrair os percentis de cor das imagens e utilizar o método conhecido como *K-Nearest Neighbor* proposto por Mitchell (1997) para criar o classificador. Apesar desse trabalho ter como objetivo classificar uma imagem apenas em duas categorias, com defeito e sem defeito os resultados foram encorajadores (KAUPPINEN, 1999).

O método de Kauppinen (1999), tábuas de Pinus com base num conjunto de seis tipos de defeitos, utilizando histogramas com bandas vermelha, verde e azul, obteve 34% de erro. Esse erro de classificação elevado foi associado à dificuldade de classificar e diferenciar alguns dos tipos de defeito, como os nós cariados.

Khoury Junior et al. (2006) desenvolveram redes neurais artificiais usando percentis das bandas vermelha, verde e azul, assim como Kauppinen (1999), para avaliação de oito defeitos de madeira serrada de eucalipto e, dependendo do tamanho de blocos da imagem, obtiveram taxas de acerto de 83,1% (blocos de 64x64 pixels) e 76,6% (32x32 pixels).

Com foco em madeira serrada de eucalipto, Gomes et al. (2008) desenvolveram um sistema para identificação de defeitos e classificação de tábuas de madeira serrada de eucalipto. Esse trabalho se beneficiou de um conjunto de bibliotecas de funções fornecidas pelos fabricantes da câmera utilizada na coleta das imagens. Para a entrada dos dados foi criado um vetor com as características extraídas a partir da análise do percentil de cor, criando em seguida um classificador estatístico bayesiano (MITCHELL, 1997). Após a primeira classificação, foi aplicado um algoritmo de ajuste que, a partir de informações morfológicas do bloco analisado, redefiniu a classificação final. O classificador foi desenvolvido com a linguagem de programação C++ e trabalhou com imagens subdivididas em blocos de 64x64 pixels (GOMES et al., 2008). Utilizando normas padronizadas por serrarias comerciais, obteve uma taxa de acerto de 81%, o que pode ser considerado um ótimo desempenho, porém quando o referencial foram as normas da ABNT, a taxa de acerto caiu para 64,3%. Essa taxa de acerto é considerada baixa, para os padrões atuais.

Oliveira et al. (2008) demonstraram que o uso de imagens em tons de cinza apresenta melhor desempenho competitivo, já que equipamentos dotados de sensores coloridos são mais caros. As imagens em tom de cinza capturadas com o auxílio de uma câmera tipo *line scan* e processadas para caracterização de cor e textura, como suavidade, aspereza e regularidade, não causaram impacto no desempenho das redes neurais SVM, utilizadas. Nesse trabalho foi desenvolvido um método para a detecção de defeitos na madeira de Pinus, a partir de imagens. O método obteve êxito na detecção de defeitos em 98,7% dos casos. Também foi experimentada a técnica MLP que, da mesma forma, apresentou 98,1% de acerto. O trabalho demonstrou a possibilidade de se trabalhar com imagens em escala de cinza e obter resultados muito próximos aos obtidos com emprego de imagens coloridas.

Rall (2010) desenvolveu um *software* em linguagem Java, para classificação de tábuas de Pinus a partir de imagens digitais. O *software* obteve uma taxa de acerto de 90,50% na classificação de uma amostra de 84 tábuas. O *software* de processamento de imagens digitais, agilizou os processos de detecção e classificação dos defeitos (nós, medula e bolsa de resina) e das tábuas, com uma alta porcentagem de acerto. Essa taxa de acerto expressiva foi obtida com o uso de várias técnicas associadas para a análise de imagens, já que, como ficou demonstrado, uma técnica isolada não se mostrou eficaz. As técnicas empregadas foram a segmentação por crescimento de regiões e por limiarização, a operação binária XOR, operações

morfológicas de abertura e fechamento para eliminação de ruídos, além de matriz de coocorrência com análise das características de entropia e contraste.

Trabalhos mais recentes aplicam sistematicamente aprendizado de máquina para classificar madeira serrada. Almeida (2014), desenvolveu *software* com modelo de classificação baseado nos defeitos consolidados e identificados na tábua de madeira serrada de Pinus, utilizando SVM e redes neurais. Na classificação dos defeitos as técnicas de SVM e redes neurais clássicas apresentaram, respectivamente, taxa de acerto de 96,88% e 94,79%. Os resultados do trabalho demonstraram que as técnicas de aprendizado de máquina aliadas às técnicas de processamento de imagens geraram resultados aceitáveis na classificação de defeitos em madeira serrada de Pinus, mas principalmente para a classificação da madeira segundo seu nível de qualidade, a fusão das técnicas apresentou acurácia satisfatória.

Vieira (2016), aplicou técnicas de aprendizado de máquina, para classificar tábuas de madeira serrada de Pinus, com base em normas comerciais de algumas serrarias. Considerou, inicialmente, três classes de qualidade, porém, ao final, agrupou-as de forma a trabalhar com apenas duas classes de tábuas. Comparou a técnica CNN com outras técnicas tradicionais como máquinas de vetores suporte, árvores de decisão, método de análise dos vizinhos próximos (KNN) e redes neurais. O método que apresentou melhor taxa de acerto (em torno de 87%) foi o descritor de texturas, usando máquina de vetores suporte (VIEIRA, 2016).

Aplicando redes neurais artificiais com retropropagação de erros, Kamal (2017) desenvolveu um modelo para classificação de nós em tábuas de madeiras, comparando resultados em duas situações, classificação quando a entrada da RNA deu-se com medidas de energia de texturas, obtendo 90,5% e classificação com entrada baseada nos valores da matriz de coocorrência do nível de cinza com 84,3% de acerto. A Tabela 3 sintetiza os trabalhos citados, comparando suas acurácias e restrições.

Tabela 3 - Trabalhos realizados na área

Ano	Autor/Data	Técnica	Espécie	Acurácia	Restrição
1989	Koivo et al.	Árvores de Decisão	Carvalho vermelho	97,2%	Detecção de defeitos, 9
1996	Lebow et al.	Funções discriminantes	Abeto	97,0%	Detecção de defeitos, 8
1999	Kauppinen	Aprendizado de máquinas com k-NN	Pinus	96,6%	Detecção de defeitos, 6
2001	Radovan et al.	Processamento de imagens Segmentação	Coníferas	83,6%	Detecção de defeitos, 6
2006	Khoury Jr	Análise multivariada	Eucalipto	97,6%	Detecção de defeitos, 12
2008	Gomes et al.	classificador estatístico bayesiano	Eucalipto	64,3%	Detecção de defeitos, 12
2008	Oliveira et al.	Redes Neurais Artificiais e SVM	Pinus	98,1% 98,7%	Existência ou não de defeitos
2009	Marcano-Cedeno et al.	Redes Neurais Artificiais	Coníferas	97,9%	Detecção de defeitos, 3
2010	Rall	Processamento de imagens, várias técnicas	Coníferas	90,5%	Classificação Tábua, sem generalização
2014	Almeida	Aprendizado de máquinas com SVM e Redes neurais	Pinus	95,0%	Classificação Tábua, somente nós
2016	Vieira	Descritor de Textura através da <i>Support Vector Machine</i>	Pinus	87,2%	Classificação Tábua, apenas duas classes
2017	Kamal et al.	Redes neurais <i>feedforward</i>	Pinus	90,5%	Classificação tipos de nós

Os avanços científicos e tecnológicos têm contribuído para o desenvolvimento de diversos equipamentos comerciais (Tabela 4). Contudo são ainda muito caros e não cabem na realidade econômica da maioria das empresas de pequeno e médio porte do setor de madeira serrada brasileiro.

Tabela 4 - Produtos disponíveis no mercado e seus recursos

Produto / Empresa	Equipamento e sensores incorporados	Recursos e Aplicações
ProfiGrade / Lisker Oy	Câmeras matriciais a laser + digitais	Mede a forma dos troncos e tábuas; Grã madeira serrada; detecta defeitos de madeira.
LuxScan / Weinig	Câmeras coloridas + Lasers + Raios-X + Sensor acústico	Detecta defeitos da placa; mede parâmetros de forma;
Wood-X / Limab Oy	Laser + Câmeras + Raio-X	Medir a dimensão da madeira serrada; medir o volume de toras; detectar defeitos de madeira.
Detector Visual de Defeitos / Autolog	Câmera CCD + iluminação LED	Mede o perfil e a dimensão da madeira serrada; detecta nós; classificação por propriedades de nó.
DynaVision / LMI	Laser 3D + Digitalização + Visão por cortina de luz	Medir a dimensão das placas e troncos; detectar defeitos; mede e classifica, toras e madeira serrada. Detecta a posição, forma e extensão dos defeitos comuns da madeira; mede a direção da fibra e largura, espessura e comprimento da madeira
WoodEye / Innovativ Vision AB	Câmera + laser	Medir a dimensão das placas e troncos; detectar defeitos; mede e classifica, toras e madeira serrada. Detecta a posição, forma e extensão dos defeitos comuns da madeira; mede a direção da fibra e largura, espessura e comprimento da madeira
NV2000 / Ventek	Camera + Iluminação	Detectar e classificar defeitos de compensado; mede a umidade.
Optiside / Microtec	Raios-X + Laser + Infravermelho + Câmera + Tomografia	Mede as dimensões e a forma das peças; detecta defeitos internos e superficiais; Avalia a força e a umidade da madeira;

Fonte: Adaptado de ÖSTERBERG (2009).

3 MATERIAL E MÉTODOS

Para avaliar a acurácia dos modelos de RNAs aplicando o *deep learning* na classificação automatizada de tábuas de *Pinus taeda* e *Pinus elliotti*, foram utilizadas as imagens de tábuas completas e blocos de defeitos extraídos dessas imagens disponibilizados por Rall (2010), Gomes (2013) e Almeida (2014).

O trabalho foi dividido em três momentos, com abordagens específicas para o problema em questão.

Na primeira abordagem desenvolveu-se um modelo de redes neurais convolucionais para classificar blocos de imagens com dimensão de 32x32 pixels, extraídos de imagens de tábuas de madeira de pinus em duas categorias: imagens que apresentam qualquer tipo e tamanho de defeito e imagens totalmente isentas de defeitos.

Na segunda abordagem outro modelo CNN desenvolvido com base nos resultados da primeira abordagem, foi utilizado para classificar blocos de imagens com dimensão de 128x128 pixels, em seis categorias estabelecidas de acordo com o tamanho da área da imagem acometida por qualquer tipo de defeito.

Na terceira abordagem, a última e principal, a classificação utilizou imagens completas de tábuas, considerando as categorias estabelecidas pela norma ABNT e que foram representadas nos conjuntos de imagens disponíveis.

A Figura 19 apresenta uma amostra de imagem de tábua completa e de blocos de imagens extraídos da imagem.

Figura 19 – Tábua completa e blocos de imagem extraídos da tábua



Essa divisão foi necessária devido às variações de técnicas e estratégias adotadas para cada conjunto específico de imagens e para que fosse possível evoluir, de forma gradual, na complexidade do modelo desenvolvido, facilitando o entendimento do seu funcionamento. Pretendeu-se assim, criar uma arquitetura para

o modelo de redes que tenha seu comportamento gerenciável em detalhes, que classifique com alto índice de acerto e, como é característica das CNNs, possa ser aproveitado em outras aplicações.

3.1 Material

3.1.1 Conjunto Principal de imagens

Como base para o desenvolvimento de todo o trabalho, foi utilizado um conjunto de imagens coletadas no trabalho de Gomes (2013), onde foram desdobradas 3 árvores de *Pinus elliottii* com aproximadamente 52 anos de idade, resultando em 81 tábuas com dimensões aproximadas de 250cm de comprimento, 2,5cm de espessura e largura variando entre 12,0cm e 30,5cm. Todas as árvores foram colhidas no Horto Florestal do município de Manduri - SP. O desdobro, que consiste na divisão da árvore em toras e, em seguida, em tabuas, e o refilamento (retirada das bordas com cascas da tábua) ocorreram na serraria do próprio local.

Como foram coletadas imagens das duas faces da tábua geraram-se, nesse primeiro momento, 162 imagens das tábuas sem nenhum processamento mecânico adicional além do refilamento. Tábuas antes e após o refilamento podem ser observadas nas Figura 20 e 21, respectivamente.

Figura 20 – Detalhe de algumas tábuas desdobradas



Figura 21 – Detalhe de uma tábua após o refilamento



Fonte: GOMES (2013)

Essas imagens representam as tábuas em seu estágio inicial, sem muito tratamento e processamento mecânico, como muitas vezes são comercializadas e deixam as serrarias para seu destino.

Um outro grupo, de mais 162 imagens, foi formado a partir da coleta das imagens das duas faces das mesmas 81 tábuas pelos processos de secagem em estufa (para um teor de umidade de 12%) e aplainamento, na mesma serraria de origem. A Figura 22 apresenta a imagem de uma tábua após esses processos.

Figura 22 – Tábua após o aplainamento e secagem



Fonte: GOMES (2013)

As imagens dos dois grupos foram recortadas de forma a subtrair as regiões que não são relevantes para o estudo como o fundo (Figura 23). O procedimento foi realizado manualmente, usando a ferramenta de corte de imagem do *software Microsoft Office Picture Manager*. Esse processo resultou em imagens de 2592 pixels de comprimento por 394 pixels de largura, com aproximadamente 350 *kylobytes*. As imagens foram geradas, originalmente, com dimensão de 2592x1944 pixels, por equipamento fotográfico com resolução máxima de 5 megapixels, gerando arquivos de tamanho aproximado de 1.2 megabytes.

Figura 23 – Imagem da tábua após recorte



Fonte: GOMES (2013)

Desse conjunto algumas tábuas foram classificadas manualmente pelo autor, conforme disposto na Tabela 5.

Tabela 5 – Classificação das tábuas de *Pinus elliotti*

Classe	Ocorrências	Porcentagem
Super	4	5%
Extra	16	20%
Primeira	38	48%
Segunda	18	22%
Terceira	4	5%
Total	80	100%

Fonte: Adaptado de GOMES (2013).

3.1.2 Conjunto Adicional de imagens

Promoveu-se a classificação de tábuas de um outro conjunto de 84 imagens obtidos do trabalho de Rall (2010), Conjunto Adicional. As imagens geradas nesse trabalho são distintas do Conjunto Principal, tanto na espécie da madeira quanto em suas dimensões, além de terem sido capturadas em ambiente com outras características de luz e resolução. O objetivo foi avaliar a capacidade de generalização do modelo, aplicando-o a outros panoramas, o que ocorre constantemente no cotidiano das serrarias e indústrias do setor.

O processo de geração das imagens encontra-se descrito em detalhes no referido trabalho. Basicamente, o Conjunto Adicional é composto por imagens de madeira serrada de *Pinus taeda* desdobradas de 6 árvores de aproximadamente 37 anos de idade, colhidas no mesmo Horto Florestal de Manduri. Todas tábuas foram refiladas e aplainadas, ficando com dimensão de 200cm de comprimento e larguras variando de 13,5 a 32,5cm.

As imagens, adquiridas com máquina fotográfica digital em sua resolução máxima de 6 megapixels, resultaram com dimensões 3072x2048 pixels, armazenadas em arquivos de aproximadamente 2 megabytes. A Figura 24 traz a amostra de uma tábua desse conjunto.

Figura 24 – Imagem da tábua de *Pinus taeda*

Fonte: Adaptado de RALL (2010).

As tábuas desse conjunto foram classificadas manualmente pelo autor, seguindo a íntegra da norma ABNT que direciona esse procedimento. O resultado dessa classificação é disposto na Tabela 6.

Tabela 6 – Classificação das tábuas de *Pinus taeda*.

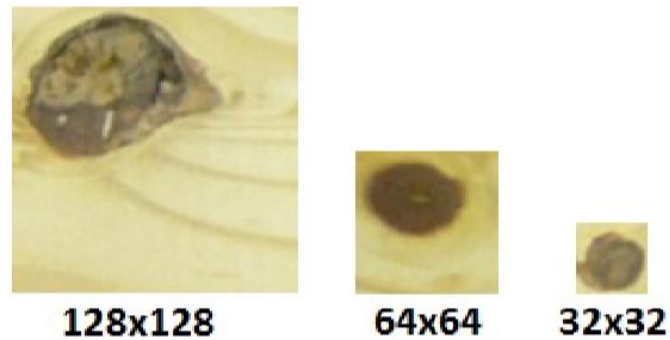
Classe	Ocorrências	Porcentagem
Super	16	19%
Extra	0	0%
Primeira	41	49%
Segunda	27	32%
Terceira	0	0%
Total	84	100%

Fonte: Adaptado de RALL (2010).

3.1.3 Grupos de imagens

As imagens utilizadas na primeira e segunda abordagem - blocos de imagens menores - foram geradas pela subdivisão das imagens de tábuas completas dos Conjuntos Principal e Adicional. Essa subdivisão foi efetuada por processo automatizado, descrito em Almeida (2014). Esse processo gerou centenas de novas imagens com resoluções de 32x32, 64x64 e 128x128 pixels (Figura 25), contendo ou não defeitos, pois foram extraídas segundo uma sequência de recorte definida previamente.

Figura 25 – Imagens de blocos e suas dimensões



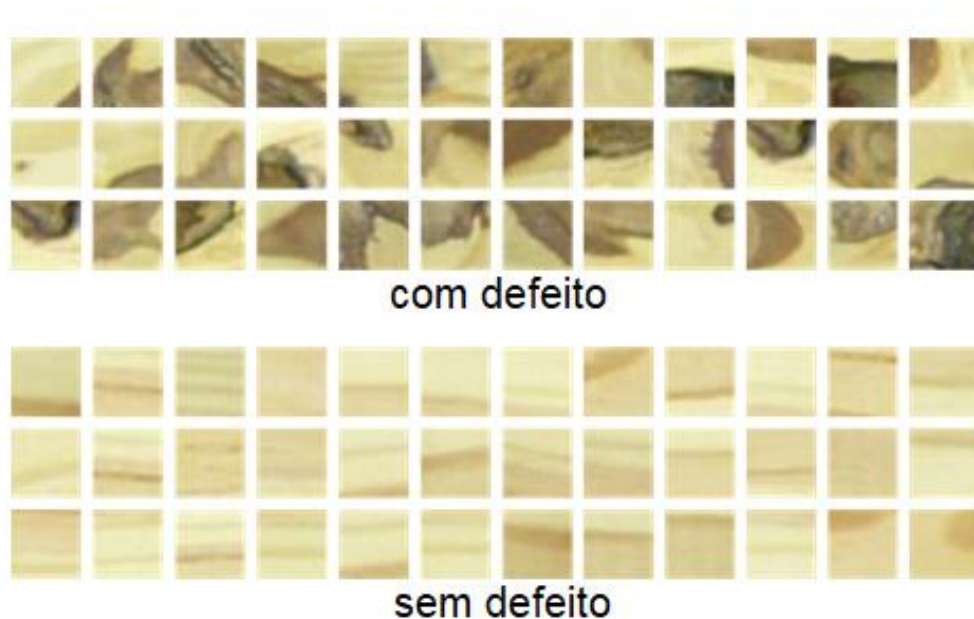
Fonte: Adaptado de ALMEIDA (2014).

3.1.3.1 Grupo 1

Esse grupo de imagens foi aplicado na primeira abordagem, para avaliar, com uso do modelo CNN a presença ou ausência de defeitos (madeira limpa) em blocos com 32x32 pixels, com intuito de verificar o comportamento e efetividade do modelo nesse panorama, mais simples.

O grupo foi formado por 440 imagens - blocos - sendo que 200 apresentam defeitos e 240 não - madeira limpa (Figura 26).

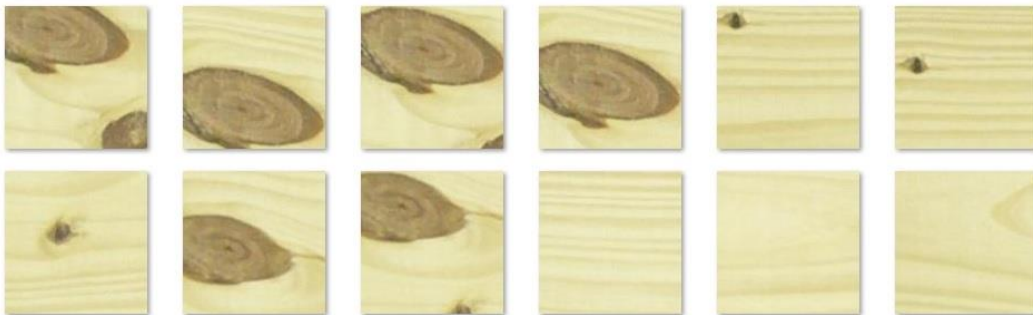
Figura 26 - Imagens de blocos com 32x32 pixels



3.1.3.2 Grupo 2

Evoluindo na complexidade do modelo de redes neurais convolucionais, na segunda abordagem foram utilizadas 343 imagens com dimensão maior - blocos de 128x128 pixels (Figura 27), extraídas do Conjunto Adicional (RALL, 2010) e classificadas previamente em seis categorias.

Figura 27- Exemplos das imagens de 128x128 pixels



A classificação foi executada por visão humana. Foram atribuídos rótulos de acordo com a porcentagem de defeitos que pôde ser observado, considerando seis categorias que variam de madeira limpa, com 0% de defeito, denominada D0, até bloco madeira que apresente entre 81% e 100% da área com defeito, denominada D5 (Figura 28 e Tabela 07). Esse processo seguiu basicamente a mesma metodologia descrita por Almeida (2014), porém com o diferencial de que aqui não contemplamos apenas nós, mas sim qualquer tipo de defeito presente na tábua, apesar de o nó ser o tipo de defeito que mais acomete a madeira de pinus e passa de 95% dos defeitos existentes nas amostras aqui estudadas.

Figura 28 - Classificação e rótulos das imagens do Grupo 2

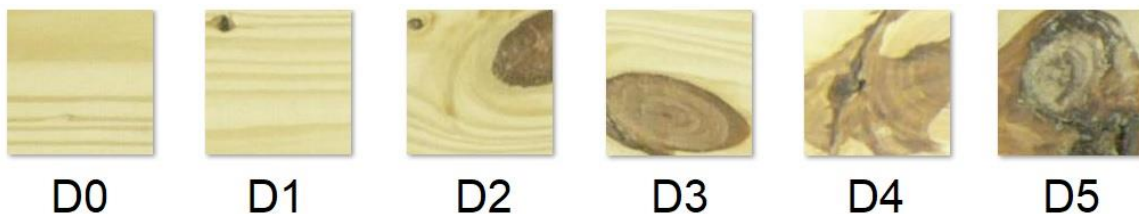


Tabela 7 – Parâmetros das classes das imagens do Grupo 2

Classe	% defeito aproximada	Rótulo	Quantidade
D0	0	1	77
D1	1 a 20	2	61
D2	21 a 40	3	102
D3	41 a 60	4	31
D4	61 a 80	5	29
D5	81 a 100	6	43
Total			343

3.1.3.3 Grupo 3

Na terceira abordagem foram usadas imagens das tábuas inteiras, para atingir o objetivo principal de classificação de tábuas de madeira serrada de Pinus nas categorias definidas pela norma.

Dentre as imagens que compõem o Conjunto Principal e o Conjunto Adicional não foram encontradas tábuas que representassem a terceira classe, definida pela norma ABNT como sendo a mais afetada por defeitos e, por consequência, a de pior qualidade comercial. Para suprir essa falta, efetuou-se a seleção e recorte de porções específicas, com dimensão de 100x1000 pixels, de algumas imagens do Conjunto Adicional, de forma que se conseguisse 43 imagens enquadradas nessa categoria.

Sendo assim, o Grupo 3 foi formado por uma mescla de imagens geradas a partir do Conjunto Principal e do Conjunto Adicional. Do Conjunto Principal foram geradas imagens com dimensão 100x1000 pixels, representativas das classes super, extra, primeira e segunda, 50 imagens para cada uma das classes, totalizando 200 imagens. A geração e classificação dessas imagens deu-se conforme descrito no trabalho de Almeida (2014) e, basicamente, consistiu em recortar imagens de tábuas completas e classificá-las segundo a ABNT. Estudos preliminares apontaram para diferenciação de tábuas das classes extra e primeira, mais complexas, exigindo uma quantidade maior de imagens. A esse conjunto de 200 imagens foram adicionadas as 43 imagens

de terceira classe já citadas e outras 41 amostras de imagens de primeira classe (Tabela 8).

Tabela 8 – Distribuição das imagens de tábua completa, por classe

Classe	Almeida (2014)	Rall (2010)	Ocorrências	Porcentagem
Super	50	0	50	17.5%
Extra	50	0	50	17.5%
Primeira	50	41	91	32.0%
Segunda	50	0	50	17.5%
Terceira	0	43	43	15.5%
Total	200	84	284	100%

Os dados de entrada para treinamento e teste dos modelos computacionais de redes neurais aqui desenvolvidas foram coletados a partir da leitura das imagens disponíveis. Para cada abordagem foram destinadas imagens com características específicas e adequadas aquela etapa, reunidas em conjuntos com a nomenclatura especificada. Para a configuração da arquitetura do modelo CNN utilizado todas as imagens estavam no padrão de cores RGB, o mais recomendado. A Tabela 9 compila os conjuntos de imagens aplicados neste trabalho.

Tabela 9 – Nomenclatura e características dos conjuntos de imagens

Nome	Total unidades	Dimensão pixels	Tipo de imagem	Origem
Conjunto Principal	324	2592x394	Completa	Gomes (2013)
Conjunto Adicional	84	3072x2048	Completa	Rall (2010)
Grupo 1	440	32x32	Bloco	Conjunto Principal
Grupo 2	343	128x128	Bloco	Conjunto Adicional
Grupo 3	284	1000x100	Recorte	Almeida (2014) e Conjunto Adicional

3.1.4 Equipamento computacional

Os modelos de redes neural da primeira e segunda abordagem desse trabalho, até determinado momento, na forma de seus algoritmos, foram executados em um computador pessoal (PC) do tipo notebook, equipado com processador da fabricante Intel modelo Core i7 4510U de quarta geração com dois núcleos físicos de 2.00 GHz de velocidade por núcleo, sendo 8GB de memória RAM.

Em dado momento, ainda na segunda abordagem, o equipamento inviabilizou a execução dos testes, elevando em muito o tempo de treinamento, por conta de suas limitações de hardware. Fez-se então necessário buscar uma plataforma alternativa, que rodasse mais rapidamente os experimentos. Sabendo-se que, em comparação às CPUs (*Central Processing Unit* - microprocessadores que equipam os computadores pessoais), as GPUs (*Graphics Processing Unit*) são, por várias razões, mais indicadas para treinamento de modelos RNAs que aplicam aprendizagem profunda e redes convolucionais. Efetuou-se então uma pesquisa por soluções que disponibilizassem, de forma gratuita, plataformas equipadas com tal tecnologia. Sendo identificada e alocada a plataforma CoLAB, também chamada *Colaboratory*.

3.1.4.1 Plataforma online CoLAB

Criada pela empresa Google para incentivar e disseminar as pesquisas relacionadas ao aprendizado de máquinas. Consiste em uma plataforma gratuita e executada totalmente em nuvem, necessitando apenas de um navegador *web* e conexão com internet. É compatível com a linguagem Python e as ferramentas e bibliotecas mais usadas nessa área, TensorFlow e Keras, desenvolvidas pela mesma empresa. O ambiente é interativo e provê suporte tanto para a escrita do código quanto a sua execução, também disponibiliza recursos de análise e compartilhamento dos resultados obtidos. A partir da autenticação na plataforma, totalmente online, pode-se treinar os modelos de redes em um equipamento de computação otimizado para esse fim, muito superior a computadores pessoais, tendo como configuração unidades de processamento gráfico GPUs, de marca NVidia modelo Tesla K80, com 24 GB de memória RAM dedicada, que processa os dados simultaneamente em 4.900 núcleos de processamento para treinamento, em comparação com os cerca de 7

núcleos de um PC. Bastando, para o uso desse ambiente de desenvolvimento e treinamento dos modelos de rede, acessar o sítio colab.research.google.com utilizando uma conta de e-mail do provedor da própria Google, o Gmail e seu repositório em nuvem, o Google Drive (COLAB, 2019).

3.2 Métodos

A carga do conjunto de dados definido (*dataset*) - conversão da imagem em um conjunto de dados numéricos que descrevam essa imagem - foi feita utilizando as funções de entrada de dados da biblioteca Keras, com base na função *load_data()* presente na biblioteca Keras. Tal função foi inicialmente construída para carregar o *dataset* do *Modified National Institute of Standards and Technology* - MNIST, que consiste em um conjunto de 70.000 imagens de números manuscritos. Por conta das características do MNIST serem semelhantes às do problema aqui abordado, as funções são adequadas para converter as imagens de madeira completa e dos blocos de imagens em dados numéricos e carregá-los como entrada nos modelos desenvolvidos.

Muitos recursos e técnicas de DL estão presentes nos modelos desenvolvidos nas abordagens do trabalho, como o achatamento, a subamostragem e as conexões do tipo todos-todos, além da própria convolução.

3.2.1 Primeira abordagem

A rede neural modelada nesse primeiro momento, foi construída para empregar a técnica de aprendizagem profunda em um panorama mais simples, tanto pela dimensão reduzida das imagens (conjunto menor de dados de entrada), como pela quantidade de classes envolvida (bloco com defeito e sem defeito).

Efetuuou-se o treinamento do modelo de redes convolucionais desenvolvido com várias configurações para um mesmo panorama, a fim de verificar e avaliar seu comportamento e a eficiência do modelo em cada configuração, ponderando também o tempo de processamento utilizado.

Do total de 440 imagens, 293 foram reservadas para o treinamento do modelo convolucional e 147 para serem utilizadas posteriormente, na fase de testes (Tabela

10). Seguindo a prática comumente aplicada quando não se tem grande quantidade de dados, de reservar 2/3 dos dados para treinamento e 1/3 para testes.

Tabela 10 – Divisão das imagens do Grupo 1

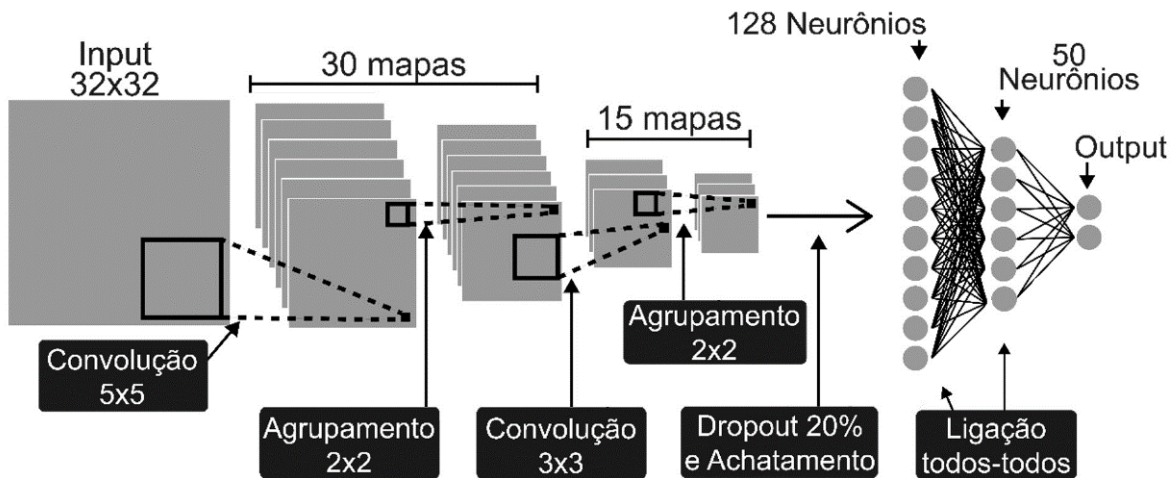
	Com defeito	Sem defeito	Total
Treino	133	160	293
Teste	67	80	147
Total	200	240	440

Para o desenvolvimento do modelo de Rede Neural Convolutiva foi utilizada a versão 2.7 da linguagem de programação Python, a biblioteca Keras e as bibliotecas de apoio TensorFlow e NumPy.

Como as imagens de 32x32 pixels usavam o padrão de cores RGB, o modelo de treinamento foi definido com 3 unidades de entrada, uma para cada banda de cor, sendo que cada unidade somou 1024 dados (32x32), totalizando 3072 inputs. Todas as imagens do conjunto de dados foram processadas pelo modelo CNN em vários ciclos de treinamento, chamados de épocas. Uma época consiste em um ciclo completo de processamento, que inclui processar os parâmetros sobre todas as instâncias do conjunto de dados reservados para treinamento.

A base para o modelo aplicado nesse trabalho foi o proposto por Brownlee (2016a). O modelo de Brownlee, denominado *larger_model*, obteve 0,89% de erro no processamento de um conjunto de dados voltado para estudos do MNIST. Basicamente, as diferenças entre o modelo *larger_model* e o aqui utilizado foram a quantidade de mapas de característica e a dimensão dos campos receptivos locais na primeira camada de convolução. Os pesos e vieses podem ser inicializados com zero ou de forma aleatória, o que é possível a partir da utilização da variável *seed* (ou semente). Essa variável tem valor definido no início do algoritmo e determina como os valores aleatórios serão gerados. No caso de aplicação o valor 7 foi fixado para *seed*, nessa e em todas as demais abordagens, sem um motivo específico para tanto. A arquitetura do modelo é representada na Figura 29.

Figura 29 - Arquitetura do modelo de rede convolucional construído - primeira abordagem



Uma das camadas do modelo realiza o chamado *dropout*, que é uma solução que propõe a não utilização de uma certa porcentagem dos *outputs* da camada anterior na próxima camada do modelo, descartando unidades escolhidas de forma aleatória e desfazendo todas as conexões de entrada e saída dessa unidade, visando prevenir o fenômeno do sobrecarga, conhecido como *overfitting*. As características dessa técnica de descarte promovem uma redução considerável na adaptação do modelo aos dados de treinamento (HINTON, et al. 2012).

O *overfitting*, por sua vez, acontece quando o modelo começa a utilizar muitos parâmetros aprendidos durante o treinamento para efetuar a categorização. Isso faz com que os resultados obtidos nas amostras de treinamento sejam muito bons, porém quando se tenta analisar um conjunto de imagens diferente do conjunto de treinamento, como o conjunto reservado para teste por exemplo, o modelo passa a produzir resultados ruins, o que diminui sua eficiência. O *dropout*, principalmente em trabalhos que envolvem visão computacional, contribui para que os resultados obtidos com outros conjuntos, que não o de treinamento, sejam melhores. Esse recurso descarta dados discrepantes, na porcentagem que estiver definida no modelo, esses dados podem representar ruídos na imagem ou características incomuns que ocorrem raramente. Os dados descartados, em tese, não impactam na eficácia da classificação, por não serem dados que descrevam regiões recorrentes nas amostras do conjunto de imagens (SRIVASTAVA et al., 2014).

O modelo aplicado usa outras camadas importantes que são as camadas de achatamento (*flatten*) e, nas últimas camadas, a estrutura densamente conectada (*fully connected*), as quais se assemelham as redes neurais artificiais mais comuns. A camada de achatamento transforma as camadas ocultas intermediárias (*hidden layers*), apresentadas na forma de matrizes multidimensionais, em estruturas lineares (vetores). Esse vetor pode ter, em sua sequência, outra camada com estrutura linear, tendo seus elementos conectados na forma todos-com-todos (*fully connect* - densamente conectados), ou pode gerar a própria camada de saída do modelo, onde o resultado é calculado e registrado (LIN; TEGMARK; ROLNICK, 2016).

Na arquitetura ora constituída (Figura 29), tem-se um modelo sequencial, onde a primeira camada é uma camada de convolução com trinta mapas de característica, campos receptivos locais com dimensão de 5x5 e um *input* tendo o formato de 32x32 pixels e três canais de cores. A segunda camada é uma subamostragem usando um filtro de dimensão 2x2 e a terceira camada é uma convolução com quinze mapas de características e campos receptivos locais de 3x3. A quarta camada é outra subamostragem, novamente com filtro de 2x2, enquanto a quinta camada estabelece um *dropout* da camada quatro para a sexta camada de 20%. A sexta camada é a camada de achatamento; a sétima contém 128 neurônios densamente conectada com a camada oito (conexão todos-todos). A camada oito possui cinquenta neurônios sendo densamente conectada à camada nove, que por sua vez é a camada de *output* que calcula a probabilidade de a imagem pertencer à cada uma das duas classes predeterminadas, dessa forma o maior valor será o considerado para predição. Esse modelo foi treinado e testado em cinquenta épocas.

Para entender como os elementos das camadas convolucionais afetam o resultado, procedeu-se da seguinte forma: foram definidas pequenas variações na camada um do modelo, com os campos receptivos locais fixos numa dimensão de 5x5, variaram-se as quantidades dos mapas de característica, 20, 30 e 40 mapas. Após três execuções do modelo, para cada valor do mapa de característica foi feito o teste no melhor modelo variando a dimensão dos campos receptivos locais em 4x4 e 6x6, também com três execuções para cada uma dessas dimensões, sabendo que já havia sido executado com dimensão de 5x5. Dessa maneira, foi possível avaliar o desempenho do modelo construído aplicando a técnica *deep learning* na categorização de dados de imagens de madeira, para registrar a configuração da rede

convolucional que apresentou melhor acurácia para então aplicar como base nas próximas etapas da pesquisa.

3.2.2 Segunda abordagem

A segunda abordagem foi uma evolução da anterior, tendo também o objetivo de testar um modelo de redes intermediário, em um panorama mais simples se comparado ao do objetivo principal da pesquisa. Aumentou-se a dimensão dos blocos de imagens e procedeu-se a classificação pela porcentagem de acometimento de qualquer defeito na madeira, elevando o grau de complexidade para avaliar o impacto dessa configuração no comportamento do modelo e nas exigências do mesmo, com relação ao equipamento computacional empregado no treinamento.

As imagens utilizadas (Grupo 2 - Tabela 7) foram classificadas por visão humana e divididas em conjuntos para treino e teste (Tabela 11).

Tabela 11 – Quantidade de imagens por classe para treino e teste, Grupo 2

Classe	Treino	Teste	Total
D0	54	23	77
D1	43	18	61
D2	72	30	102
D3	22	9	31
D4	21	8	29
D5	32	11	43
Total	244	99	343

Nesta abordagem criou-se um modelo de redes neurais convolucionais baseado nas experiências do modelo anterior. As técnicas para coleta e conversão dos dados de entrada, as ferramentas, bibliotecas e demais tecnologias aqui empregadas permaneceram as mesmas, como a linguagem de programação Python, as bibliotecas TensorFlow, Keras e NumPy.

Mantendo-se o padrão de cores RGB com as imagens de 128x128 pixels foram gerados 16.384 dados por banda de cor para cada unidade, totalizando 49.152 entradas no modelo.

Conforme já comentado foi usado a plataforma computacional em nuvem CoLAB, com processadores específicos para treinamento de redes neurais,

conhecidos como - GPUs. Paralelamente o treinamento foi executado na plataforma original - PC - para fins comparativos de desempenho.

Diversas configurações do modelo foram executadas, a maioria com a execução de *dropout*, porém o modelo apresentou resultados melhores com a aplicação de outra técnica de regularização, a L2. Optou-se por fim por não utilizar a técnica *dropout*, sendo que, com exceção da última camada do modelo e das camadas de achatamento e *maxpooling*, todas as camadas que possuem parâmetros treináveis foram submetidas a regularização do tipo L2, executando a função *l2_regularization*, para reduzir o *overfitting*.

A configuração do modelo que apresentou a melhor performance nesta abordagem, possui as seguintes características: 5 camadas convolucionais, 5 camadas de subamostragem com *maxpooling* posicionadas logo após cada camada convolucional, uma camada de achatamento (*flatten*) e três camadas densas de conexão do tipo todos com todos, conhecida também como *fully connected* (Figura 30).

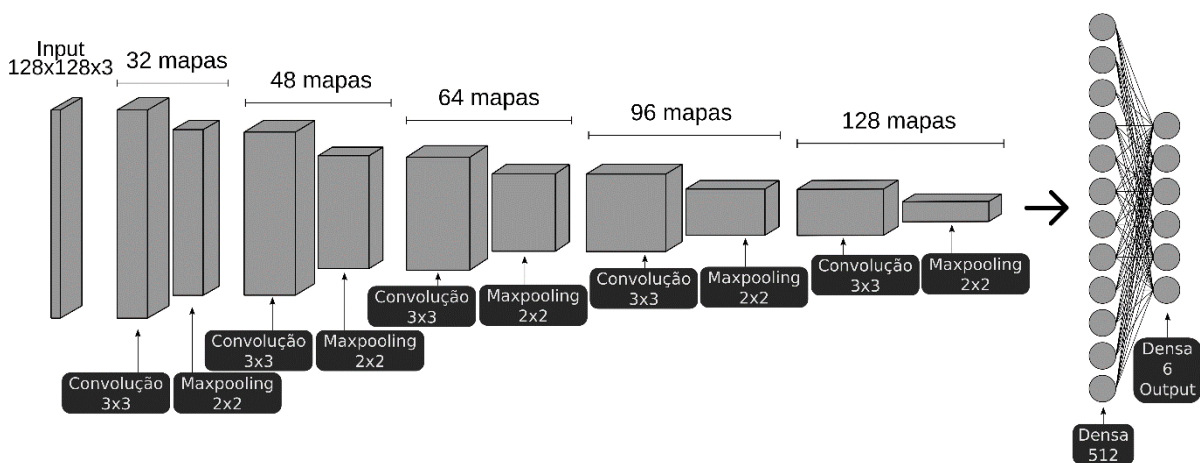
Para não perder dados nas camadas de entrada aplicou-se o *padding*, com borda de tamanho dois e valor definido em *same*, o que usa o zero como valor para o preenchimento. Neste modelo a subamostragem deu-se somente nas camadas de *maxpooling*, que preserva os neurônios mais relevantes. O tamanho do filtro (*kernel size*) usado foi de 3x3 com passo 1, para todas as camadas de convolução. Todas as camadas com parâmetros treináveis utilizaram a função de ativação ReLU, com exceção da última camada que executa ativação aplicando a função *softmax*, a qual se destina a entregar as probabilidades de a imagem pertencer à cada uma das classes.

O achatamento foi executado como penúltimo processo, antes das camadas *fully connected*, transformando a matriz de saída de três dimensões, 4x4x128, em um vetor de 2048 posições.

Na Figura 31 está representada a arquitetura do modelo de melhor performance, obtido após muitas modificações nos hiperparâmetros da rede. Pode-se verificar essas camadas, referenciadas como Conv2D, MaxPolling2D, *Flatten* e *Dense*. O *dataset* registra o total de dados de entrada no modelo e os valores em input e output representam, respectivamente, a dimensão da imagem na entrada e na saída de cada camada. O valor “*none*”, que antecede as dimensões das camadas, é o índice da imagem no lote de treinamento (*batch*); definindo-o como *none*, o preenchimento

desse índice fica por conta do modelo. O modelo divide as imagens em lotes menores que o conjunto total de dados a ser treinado, para efetuar o processo de forma ágil e gerenciável, ou seja, uma época não ocorre de uma só vez com todas as imagens sendo treinadas, nem com apenas uma imagem por vez, mas sim em lotes de imagens, definidos no modelo pela função *batch_size* (BERGSTRA; BARDENET; BENGIO; KÉGL, 2011). Na última camada do modelo, camada de saída, são entregues os valores de predição para cada imagem do conjunto. Neste modelo essa camada possui uma matriz de 244 linhas (imagens de treino) por 6 colunas (classes de defeito). Para que os valores tenham mais significado foi aplicada a função de ativação *softmax*, que normaliza os valores gerados para seus equivalentes entre 0 e 1, gerando uma distribuição de probabilidade, indicando a chance de cada imagem pertencer a uma determinada classe. Apesar de ser considerada uma função de ativação, em redes neurais profundas ela é comumente aplicada na camada da saída, para normalização (GOODFELLOW; BENGIO; COURVILLE, 2016).

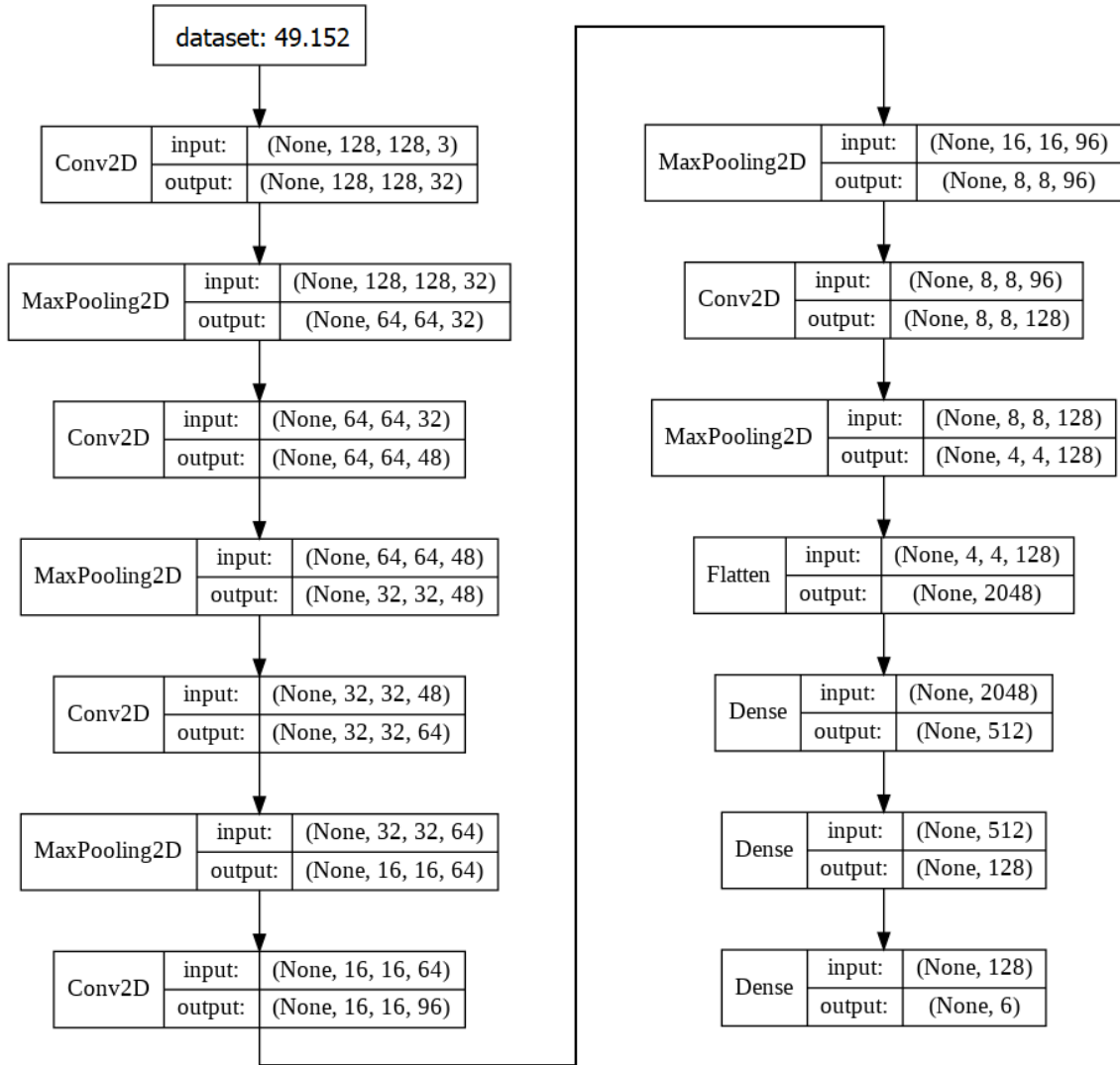
Figura 30 - Modelo de rede convolucional utilizado - segunda abordagem



Para melhorar o treinamento do modelo em questão foi empregado também o algoritmo de otimização Adam, o mais aplicado em modelos para classificação de imagens. Ele é acompanhado da definição do valor para *learning rate* (LR), que ao diminuir o tamanho do passo evita o *overshooting*. Definiu-se uma *learning rate* inicial de 0,001 e taxa de decaimento (*decay*) de 0,00013, essa taxa determina como a LR diminui ao longo do tempo. Além disso, foi utilizada uma ferramenta para redução da LR do Keras chamada *ReduceLROnPlateau*, configurada para monitorar o erro no

conjunto de treino e diminuir a LR em 15%, sempre que o modelo não apresentar diminuição no erro após 5 épocas de treinamento.

Figura 31 - Fluxo de camadas do modelo



Cada experimento foi treinado em 500 épocas com *batch* - subconjunto de imagens - de 32 imagens. Foram aplicadas algumas operações nas imagens de treinamento visando realizar transformações sutis - *image augmentation* - com o objetivo de aumentar a quantidade de amostras para alimentar o *dataset*, favorecendo a generalização do modelo e diminuindo a chance de ocorrência de *overfitting*. As técnicas de *data-augmentation* aplicadas no conjunto de imagens foram a normalização da imagem, que consiste na divisão dos valores representativos de cada pixels por 255, o cisalhamento com definição da variável aleatória entre 0 e 0,05, a rotação com variável aleatória entre 0° e 45° e o *flip* horizontal aleatório, sendo que a

cada época a função gera novos valores mantendo a aleatoriedade, criando novas imagens com sutis diferenças para o olhar humano porém com relevante impacto para o aprendizado do modelo.

3.2.3 Terceira abordagem

A terceira abordagem compreende a classificação das imagens do Grupo 3, tábuas completas de madeira serrada de Pinus, e não apenas de blocos extraídos das mesmas.

Todos os experimentos foram executados na plataforma de GPUs online CoLAB, considerando-se a maior complexidade do problema.

Promoveram-se muitas alterações na arquitetura dos modelos para atendimento as orientações da norma brasileira NBR 12297 (1991b), no tocante aos tipos de defeitos (nó, esmoado, medula, entre outros) e as cinco classes (super, extra, primeira, segunda e terceira).

Diferentemente da abordagem anterior e devido a pequena quantidade de imagens disponíveis, as imagens foram divididas em 80% para treino e 20% para teste, privilegiando o treinamento que é o processo mais importante (Tabela 12).

Tabela 12 – Divisão das imagens do Grupo 3

Classificação	Treino	Teste	Total
Super	40	10	50
Extra	40	10	50
Primeira	73	18	91
Segunda	40	10	50
Terceira	35	8	43
Total	228	56	284

3.2.3.1 Modelos de rede convolucional aplicados

Nesta abordagem foram testados dezenas de modelos. Em alguns desses modelos foram aplicadas técnicas que, apesar de não terem atingido resultados satisfatórios naquele momento, motivaram mudanças de estratégia e merecem

registro. Tais modelos estão apresentados em ordem crescente de performance, adotando-se a terminologia: modelo Adaptado, Ajustado, Simplificado e Final.

3.2.3.1.1 Modelo Adaptado

O primeiro modelo foi uma adaptação direta do modelo final desenvolvido na abordagem anterior, promovendo-se apenas os ajustes relativos a formato de entrada do *dataset* e de saída, modificando-se a dimensão dos inputs do modelo de 128x128 para 100x1000, reduzindo-se o tamanho do mini lote para 16 e taxa de aprendizado para 0,0005, mantendo-se o esquema de camadas (Tabela 13) e também alguns dos hiperparâmetros, permanecendo com total de 1500 épocas, decaimento de 0,0009 e aplicando-se o algoritmo de otimização Adam, manipulando um total de 6.370.261 de parâmetros treináveis, número que está diretamente ligado a complexidade do modelo.

Tabela 13 - Arquitetura do modelo Adaptado

Tipo Camada	Formato de saída	Parâmetros
conv2d_1	(None, 100, 1000, 32)	896
max_pooling2d_1	(None, 50, 500, 32)	0
conv2d_2	(None, 50, 500, 48)	13872
max_pooling2d_2	(None, 25, 250, 48)	0
conv2d_3	(None, 25, 250, 64)	27712
max_pooling2d_3	(None, 12, 125, 64)	0
conv2d_4	(None, 12, 125, 96)	55392
max_pooling2d_4	(None, 6, 62, 96)	0
conv2d_5	(None, 6, 62, 128)	110720
max_pooling2d_5	(None, 3, 31, 128)	0
flatten_1	(None, 11904)	0
dense_1	(None, 512)	6095360
dense_2	(None, 128)	65664
dense_3	(None, 5)	516
Total de parâmetros treináveis gerados		6.370.261
Tempo total de treinamento (hh:mm:ss)		01:56:21

3.2.3.1.2 Modelo Ajustado

Nesse modelo aplicou-se a técnica de aumento de *dataset* ou *image augmentation*. Foi utilizado inicialmente a técnica nativa do Keras, chamada *flip*, que espelha a imagem vertical e horizontalmente, pois ela não gera bordas no interior das imagens resultantes.

Como a aplicação dessa técnica não gerou melhoria significativa optou-se por criar uma função que executasse, de forma gerenciada, o recorte e o remonte da imagem. A função, que recebeu o nome de *custom_augmentation*, executa alguns processos da função *image_umentation* nativa do Keras, com a especificidade de cortar a imagem de forma gerenciada, cortando blocos horizontalmente de 100x100 pixels, gerando 10 blocos para cada imagem, os quais são reagrupados formando uma nova imagem de 1000x100. A imagem gerada apresenta características inexistentes nas imagens originais, como o limite de um bloco para outro, ressaltado pela mudança brusca no padrão de cores dos pixels, principalmente nas regiões que contém defeitos (Figura 32).

Figura 32 - Imagem após *custom_augmentation* técnica *crop* 100x100



Esse modelo, com aplicação da função *custom-augmentation*, obteve um desempenho próximo ao do modelo anterior, mas ainda insatisfatório e, em alguns experimentos, chegou a ponto de piorar o desempenho do modelo. Também foi definida a ativação ReLu como sendo uma camada, antes de cada camada de *pooling*.

Neste modelo aplicou-se uma camada de *dropout* de 20% logo após a camada de *GlobalAveragePooling* (Tabela 14). A função do Keras *Global Average Pooling* (GAP), ou agrupamento por média global, minimiza o *overfitting* por meio da redução do número de parâmetros treináveis do modelo. Esse recurso trabalha como o *maxpooling*, porém de forma mais extrema, reduzindo as dimensões espaciais de uma matriz tridimensional. Uma camada de GAP reduz cada mapa de características em um único valor, simplesmente calculando a média de todos os valores do mapa (ZHOU, et al. 2016). Ao aplicar-se o *dropout* logo após um GAP reduz-se ainda mais

a quantidade de parâmetros, minimizando a probabilidade de *overfitting*, porém correndo-se o risco do descarte de valores importantes para o aprendizado.

Tabela 14 - Arquitetura do Modelo Ajustado com melhores resultados

Tipo Camada	Formato de saída	Parâmetros
conv2d_1	(None, 100, 1000, 32)	896
activation_1	(None, 100, 1000, 32)	0
max_pooling2d_1	(None, 50, 500, 32)	0
conv2d_2	(None, 50, 500, 64)	18.496
activation_2	(None, 50, 500, 64)	0
max_pooling2d_2	(None, 25, 250, 64)	0
conv2d_3	(None, 25, 250, 128)	73.856
activation_3	(None, 25, 250, 128)	0
max_pooling2d_3	(None, 12, 125, 128)	0
conv2d_4	(None, 12, 125, 512)	590.336
activation_4	(None, 12, 125, 512)	0
max_pooling2d_4	(None, 6, 62, 512)	0
conv2d_5	(None, 6, 62, 1024)	4.719.616
activation_5	(None, 6, 62, 1024)	0
global_average_pooling2d_1	(None, 1024)	0
dropout_1	(None, 1024)	0
dense_1	(None, 512)	524.800
activation_6	(None, 512)	0
dense_2	(None, 5)	2.565
Total de parâmetros treináveis gerados		5.930.565
Tempo total de treinamento (hh:mm:ss)		01:53:50

Mantiveram-se as configurações para os hiperparâmetros, a opção pelo algoritmo de otimização Adam e valores já consolidados nos experimentos passados (Tabela 15).

Tabela 15 - Hiperparâmetros e valores definidos no experimento do modelo Ajustado

Parâmetros	Valor
Épocas	1500
Tamanho do lote	16
Tipo de otimização	Adam
Taxa de aprendizagem LR	0,0005
Taxa de decaimento	0,0009

3.2.3.1.3 Modelos Simplificado

Como os modelos anteriores atingiram uma complexidade preocupante, com quantidade de épocas elevada, aumentando em muito o tempo de treinamento e não convergiram, partiu-se para o emprego de modelos mais simples, com poucas camadas e poucos mapas de características, usando combinações de camadas e técnicas ainda não testadas (Tabela 16).

Chegou-se a um modelo simplificado, com resultados aceitáveis e com tempo de treino relativamente baixo. A principal estratégia adotada foi combinar a função *GlobalAveragePooling* e a camada *Dropout* visando reduzir a importância da localização dos defeitos no espaço da imagem. Etapa intermediária, com uso da função *BatchNormalization* entre a função *GlobalAveragePooling* e a camada *Dropout*, não teve boa performance, confirmando que o *BatchNormalization* não se comporta bem quando acompanhado da camada *Dropout* (XIANG et al, 2018).

O diferencial do presente modelo, que visa estabelecer uma arquitetura mais simples, está na redução da complexidade por meio da diminuição da quantidade de camadas e mapas de características em cada camada, que leva a um modelo que não considera, no seu processo de aprendizado, os detalhes muito específicos do conjunto de treino, produzindo um modelo mais genérico e com menor *overfitting*. Essa redução de complexidade em prol de um modelo que desconsidera características mais específicas do conjunto de dados durante o treino, é basicamente o mesmo que se tenta alcançar quando emprega-se o *dropout* juntamente com a regularização do tipo L2.

Tabela 16 - Arquitetura da melhor variação do modelo Simplificado

Tipo Camada	Formato de saída	Parâmetros
conv2d_1	(None, 100, 1000, 16)	448
activation_1	(None, 100, 1000, 16)	0
max_pooling2d_1	(None, 50, 500, 32)	0
conv2d_2	(None, 50, 500, 32)	4.640
activation_2	(None, 50, 500, 32)	0
max_pooling2d_2	(None, 25, 250, 32)	0
conv2d_3	(None, 25, 250, 32)	9.248
activation_3	(None, 25, 250, 32)	0
max_pooling2d_3	(None, 12, 125, 32)	0
conv2d_4	(None, 12, 125, 32)	9.248
activation_4	(None, 12, 125, 32)	0
max_pooling2d_4	(None, 6, 62, 32)	0
conv2d_5	(None, 6, 62, 128)	36.992
activation_5	(None, 6, 62, 128)	0
global_average_pooling2d_1	(None, 128)	0
dropout_1	(None, 128)	0
dense_1	(None, 32)	4.128
activation_6	(None, 32)	0
dense_2	(None, 5)	165
Total de parâmetros treináveis gerados		64.869
Tempo total de treinamento (hh:mm:ss)		01:45:46

O modelo difere dos anteriores na quantidade, tipo e sequência de camadas aplicadas, além de ter redução expressiva no número de *features* em cada camada. Sua simplicidade em relação aos anteriores pode ser avaliado pelo total de parâmetros treináveis gerados (64.869) contra cerca de 5 a 6 milhões dos anteriores.

O modelo teve eficiência aceitável, tempo de treinamento muito abaixo do que se tinha nos modelos ajustados e uma quantidade de camadas bem inferior à empregada até então.

3.2.3.1.4 Modelo Final

Foi o modelo que gerou a melhor performance e sofreu mais ajustes, mas nem por isso apresentou-se como sendo o modelo mais complexo ou o que mais exigiu do sistema computacional.

Nesse modelo, em especial, destinou-se algum tempo em várias experimentações com diferentes valores para os parâmetros da regularização L2. Observou-se que aplicar valores baixos no parâmetro lambda (λ) que define a intensidade da regularização efetuada pelo L2, levou a modelos com melhor performance. O uso de L2 em redes convolucionais já mostrou ser um recurso interessante para melhorar a performance em outros estudos, por isso tentou-se tais experimentações, obtendo êxito.

Na variação deste modelo Final que obteve melhor performance, foi aplicado o *Batch Normalization* nas camadas convolucionais, a *Global Average Pooling* após a última camada convolucional como subamostragem e um *Dropout* de 50% nas duas camadas *Dense*, densamente conectadas.

Deste modo os valores de L2 precisam ser cuidadosamente selecionados de forma a reduzir o *overfitting*, sendo elevados suficientemente, mas não ponto de atrapalhar a performance do modelo. Todas as camadas convolucionais utilizaram o recurso de Regularização L2, sendo configurados os valores do lambda nos patamares entre 0,001 e 0,005.

Aplicaram-se valores menores de L2 nas primeiras camadas convolucionais, buscando-se capturar mais detalhadamente características relevantes da imagem, elevando esses valores conforme a profundidade do modelo aumenta, objetivando-se evitar a priorização de características existentes somente nas imagens de treino. Esses valores foram os que melhor se adequaram ao presente problema, sem atrapalhar o modelo e reduzindo satisfatoriamente o *overfitting*. A arquitetura do modelo é apresentada na Figura 33, onde pode-se verificar as particularidades do modelo.

Aplicou-se a mesma configuração para hiperparâmetros e valores do modelo anterior, incluído a LR. O modelo gerou quantidade relativamente baixa de parâmetros treináveis, considerando modelos com arquiteturas semelhantes mais complexas,

repetindo o valor para parâmetros não treináveis devido ao uso da normalização por lotes (Tabela 17).

Figura 33 - Arquitetura do modelo Final

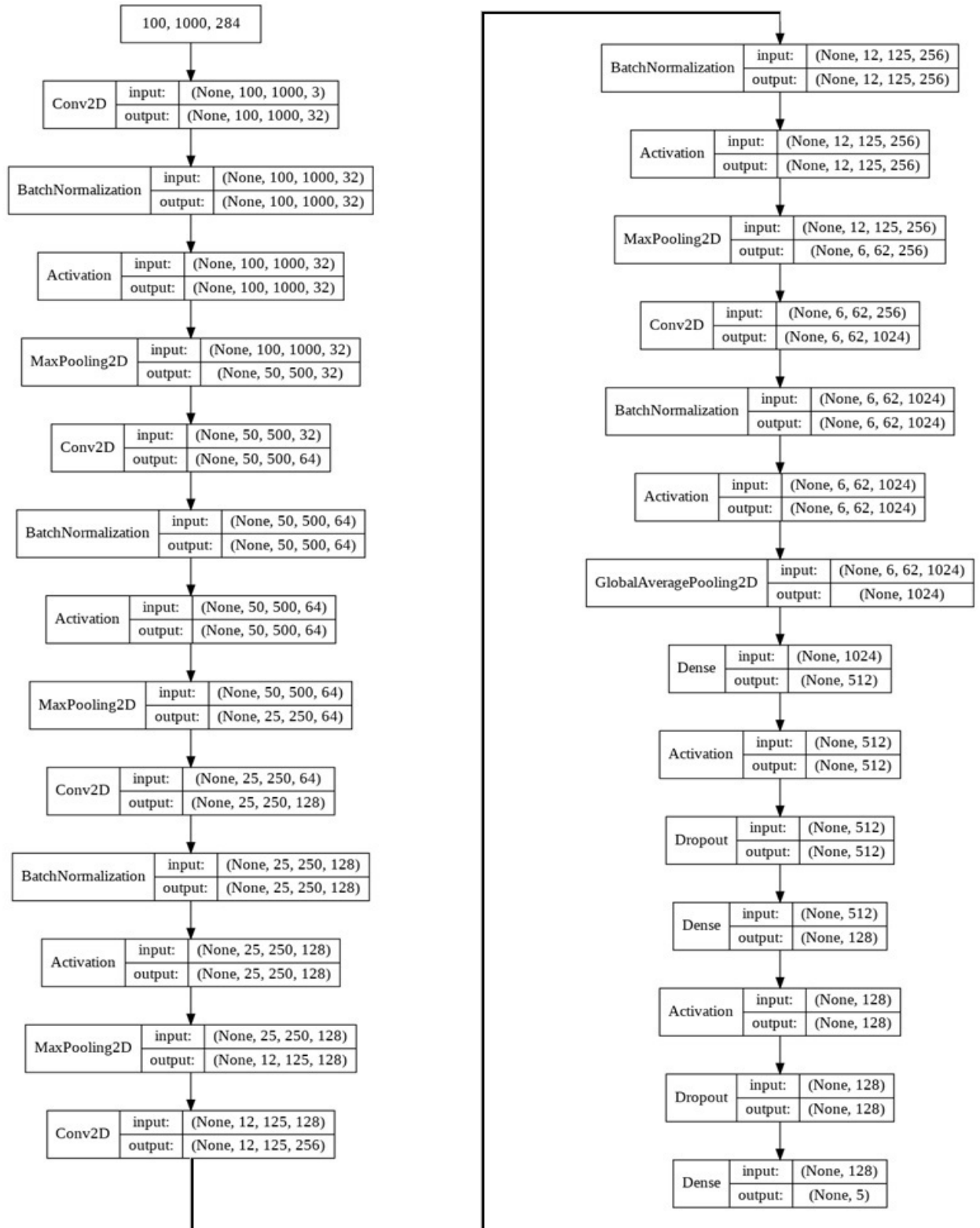


Tabela 17 - Configuração do modelo Final

Parâmetros	Valor
Épocas	1500
Tamanho do lote	16
Tipo de otimização	Adam
Taxa de aprendizagem LR	0,0001
Taxa de decaimento	0,0009
Parâmetros treináveis	3.342.853
Parâmetros treináveis	3.008
Tempo de Treino (hh:mm:ss)	02:00:29

3.2.4 Função de validação com saída formatada

Definido o modelo de maior eficiência, ou seja, menor *loss* juntamente com acurácia mais próxima possível do 100% deve-se criar uma ferramenta que receba uma imagem, execute o modelo com esse *dataset* específico, classifique a imagem e produza uma saída formatada.

A função construída promove a classificação de 10 imagens de tábuas em apenas 1 segundo. Seu uso baseia-se na execução do algoritmo que aplica modelo CNN com os parâmetros já treinados, podendo ser executada em qualquer equipamento computacional e, possivelmente, integrando um equipamento automatizado de classificação de tábuas de madeira serrada de pinus, em linha de produção.

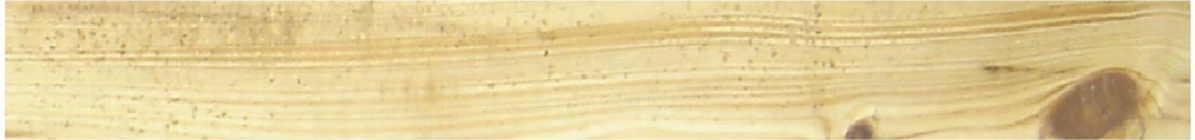
Foram definidos alguns parâmetros para seu funcionamento. O parâmetro *image_file* recebe uma *string* com o caminho para o arquivo da imagem gravado em disco, sendo assumido o nome do diretório desse arquivo como categoria a que a imagem pertence; o parâmetro *model_file* recebe o nome e caminho do modelo a ser usado para a classificação da imagem, fazendo referência a um modelo também gravado em disco.

A imagem é carregada, normalizada e uma dimensão é adicionada para informar o tamanho do batch inserido no modelo (no caso 1, sendo o formato da imagem inicialmente de [100,1000,3] para ser [1,100,1000,3]). A imagem é analisada pelo modelo que faz a classificação. O resultado é exibido, composto por uma amostra da imagem, a predição, o valor real, o resultado (correto ou incorreto) e o caminho do arquivo da imagem. Definiu-se esse formato para saída da função com intuito de

possibilitar a verificação de quais imagens impõem maior dificuldade de classificação para o modelo (Figura 34).

Figura 34 - Formato de saída da função de predição

Classificação do modelo: Segunda | Classificação real: Extra | Resultado: Incorreto | Arquivo: Extra/1_5_C1_1_A_EXT.jpg



4 RESULTADOS E DISCUSSÃO

4.1 Primeira abordagem

Conforme especificado inicialmente, foram analisadas três arquiteturas 40,30 e 20 mapas de características, todas com campos receptivos locais de dimensão 5x5.

Para o modelo aplicado, os percentuais de erro da fase de teste de cada arquitetura variam consideravelmente (Tabela 18), embora todos resultados obtidos possam ser considerados satisfatórios.

Tabela 18 - Resultado dos modelos com filtro de 5x5

Qtd. de mapas	Percentual de Erro			Tempo minutos
	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

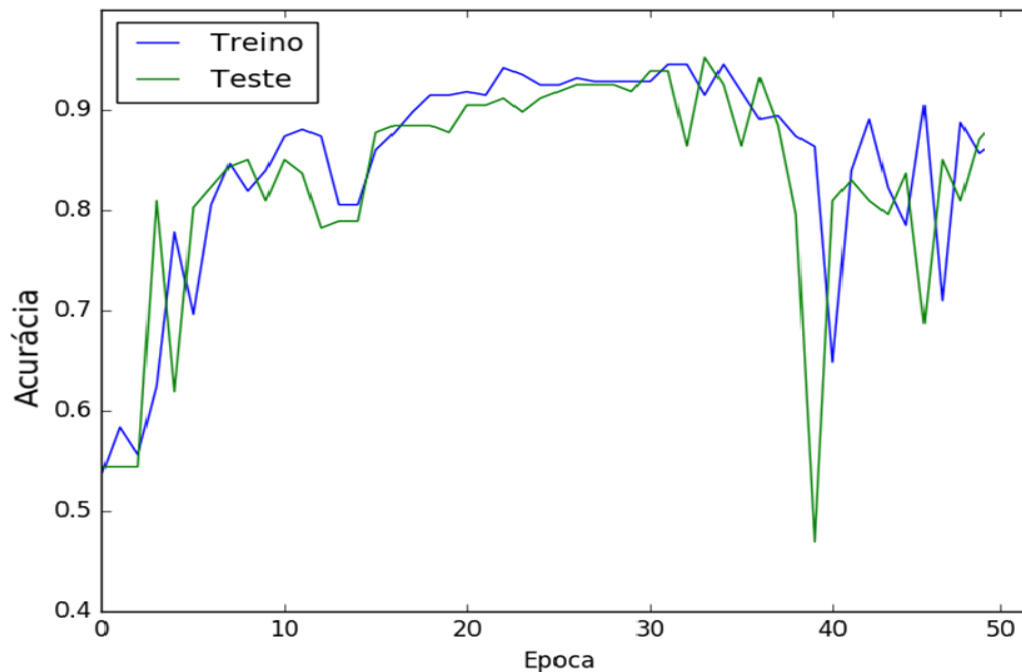
Apesar do modelo proposto ter apresentado resultados satisfatórios, não se pode afirmar que o mesmo modelo aplicado às imagens completas produzirá resultados com acurácia semelhante.

Os resultados mostraram que o aumento da quantidade de mapas de característica na primeira camada gera resultados menos satisfatórios. Isso se dá, provavelmente, devido à pequena quantidade de características que são realmente úteis e relevantes para a categorização correta. Com isso, o aumento de mapas de característica força o modelo a analisar características não relevantes para a categorização das imagens de madeiras com ou sem defeito, fazendo com que o modelo aprenda erroneamente quais características são importantes, gerando uma porcentagem maior de erros.

Pode-se perceber variação discrepante nos resultados dos experimentos com mais de 20 mapas de característica. A Figura 35 apresenta resultados obtidos na primeira execução do modelo com 40 mapas de característica, tendo como métrica a acurácia em relação às épocas. É possível 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.

Figura 35 - Acurácia em cada época do modelo de 40 mapas e filtro de 5x5



Outro fator observado foi que nos modelos treinados, quanto maior a quantidade de mapas de característica sendo utilizados pelo modelo, maior é o tempo necessário para o treinamento.

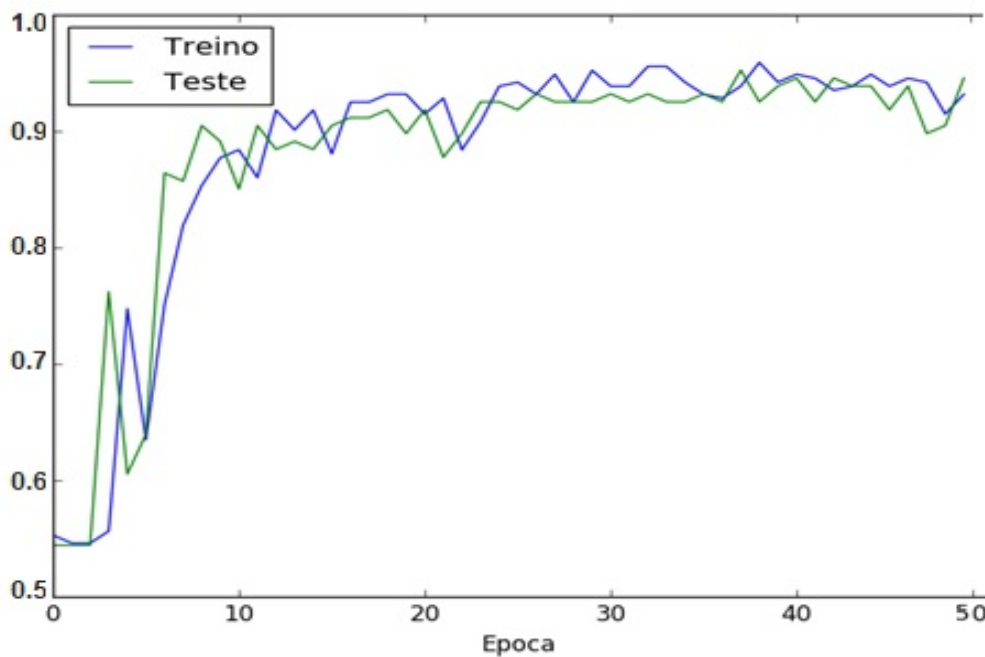
Considerando-se que o modelo com 20 mapas de característica obteve melhores resultado nas três execuções de amostra (5,44%), ele foi o escolhido para a realização dos novos testes. Foram variadas as dimensões dos campos receptivos locais para 4x4 e 6x6, e os resultados obtidos podem ser observados na Tabela 19.

Tabela 19 - Resultados dos modelos de 20 mapas de característica e dimensão variada nos filtros

Dimensão	Percentual de erro			Tempo minutos
	1ª exec.	2ª exec.	3ª exec.	
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. Isso não é uma regra para todos os conjuntos, porém as imagens utilizadas são de 32x32, fazendo com que a região captada nos campos receptivos não aponte adequadamente as características desejadas. Pelo mesmo motivo os resultados foram melhores com campos na dimensão de 4x4 e variaram menos, sugerindo maior estabilidade do modelo (Figura 36).

Figura 36 - Acurácia em cada etapa do modelo de 20 mapas e filtro de 4x4



O modelo de melhor performance - 20 mapas, filtro 4x4 - obteve 94,56% de acurácia, no trabalho de Oliveira et al. (2008), aplicaram-se RNA e SVM a um conjunto de dados um pouco maior, atingindo melhores resultados. Sabe-se que para conjuntos com poucas imagens a técnica SVM é mais adequada, porém *deep learning* tende a melhorar consideravelmente quando submetido a grandes conjuntos de dados. Vieira (2016) aplicou SVM e CNN em conjunto com 374 imagens, obtendo resultados que confirmam essa prática (Tabela 20).

Não há consenso sobre quantidade ideal para aprendizado de máquinas, a regra é quanto mais e mais representativo, melhor. Sendo assim, 500 imagens para SVM considera-se bom, para DL insuficiente.

Almeida (2014), também obteve bons resultados com SVM (96,88%) e RNA (94,79%) para um conjunto de dados semelhante ao experimentado neste trabalho, contudo não cabe comparação direta pois usou 6 classes.

Tabela 20 - Comparativo de classificadores para 2 classes

Autor	Técnica	Imagens	Acurácia
Oliveira (2008)	RNA (MLP)	500	98,10%
	SVM		98,70%
Vieira (2016)	SVM	374	98,90%
	CNN		83,00%
Autor	CNN	440	94,56%

4.2 Segunda abordagem

4.2.1 Modelo inicial

O ponto de partida para o modelo ora desenvolvido foi o modelo construído na primeira abordagem, com alterações nas funções de entrada e saída dos dados. O experimento foi executado no PC, chegando a quinquagésima época em pouco mais de treze minutos, apresentando os melhores resultados na última época (Tabela 21). Como era esperado, esse modelo inicial não convergiu, dada as características distintas do problema nesta segunda abordagem. As Figuras 37 e 38 demonstram a dificuldade do modelo inicial no início do treinamento, mas também indicam que os valores de *loss* e acurácia ainda estavam evoluindo, levando prever que, com alguns ajustes e com um número maior de épocas, o modelo poderia convergir.

Tabela 21 - Resultados do modelo inicial

Época	Parâmetro	Treino	Teste
50	Loss	0,62704	0,5003
	Acurácia	81,13%	86,86%
Tempo total de treinamento			00:13:57

Esse comportamento do modelo já era esperado devido ao conhecimento de que modelos com poucas amostras e muitas classes não convergem com poucas épocas de treinamento e necessitam de algumas técnicas para aumento de dados e otimização do modelo.

O melhor modelo é o que apresenta menor *loss* e maior acurácia na fase de teste. Contudo, geralmente um modelo não apresenta os melhores valores para *loss* e acurácia na mesma época. Nesses casos opta-se pelo modelo de menor *loss* acompanhado de acurácia aceitável, em comparação com os resultados dos demais modelos experimentados.

Apesar da acurácia de 86% ser considerada aceitável se comparada a resultados de modelos de classificação de outras pesquisas (64% e 81% Gomes et al. (2008), 76% Kauppinen (1999) e 83% Khoury Junior et al. (2006)), verifica-se na matriz de confusão dos resultados do experimento (Tabela 22), que são muitos os erros na classificação efetuada pelo modelo inicial.

Figura 37 - Evolução do *loss* segunda abordagem - modelo inicial

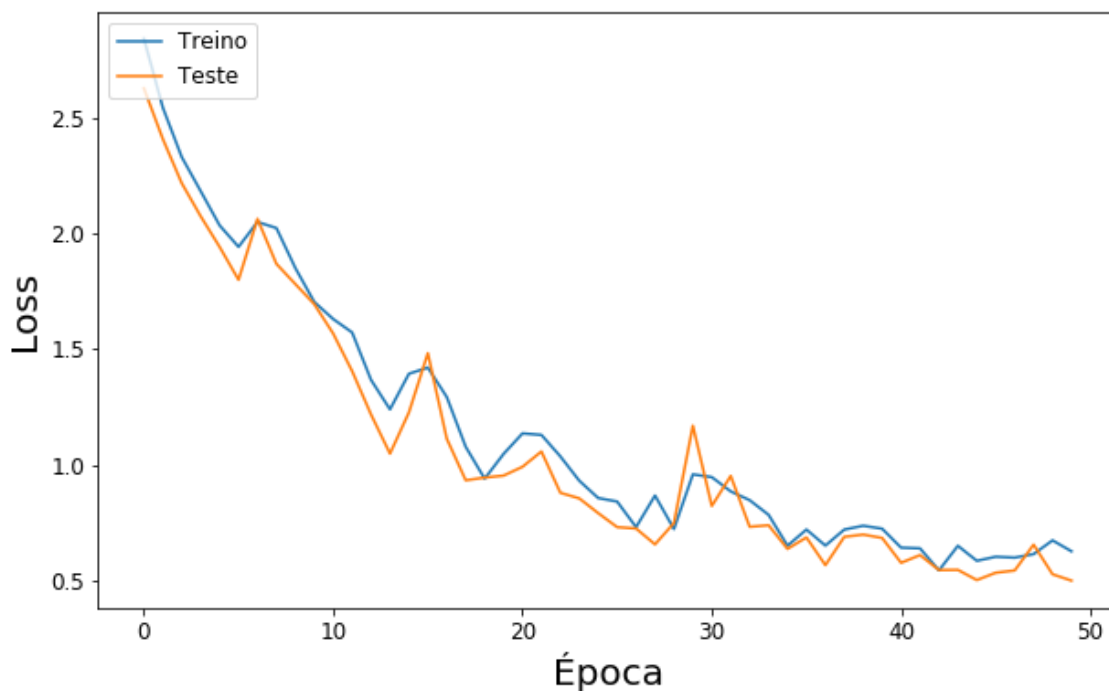
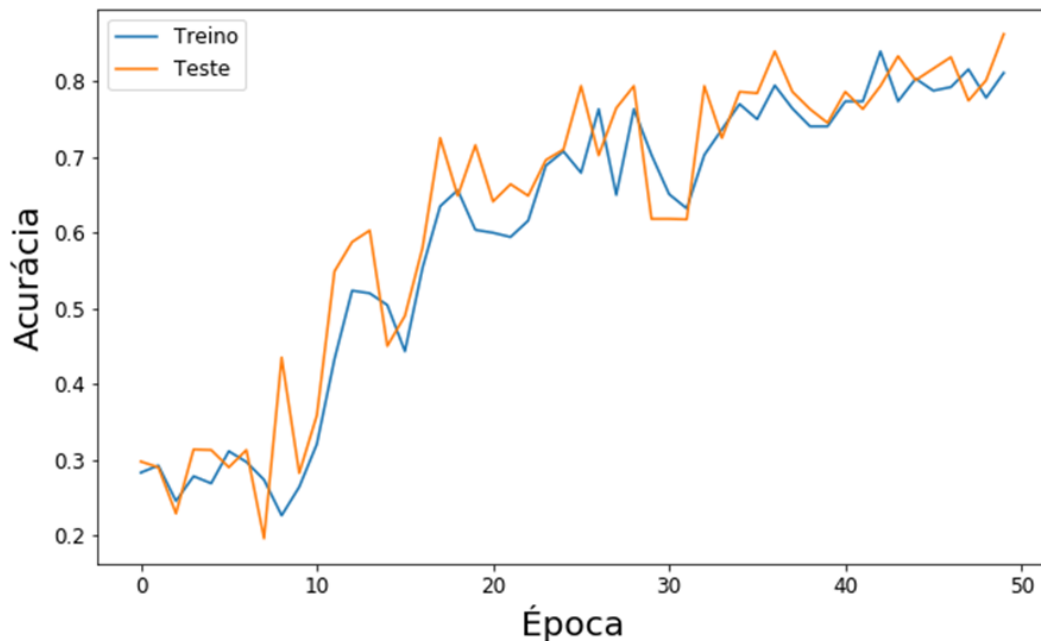


Figura 38 - Evolução da acurácia na segunda abordagem - modelo inicial**Tabela 22 - Matriz de confusão da classificação 128x128, pelo modelo inicial**

		Classificação Automatizada						
		D0	D1	D2	D3	D4	D5	
Visão humana	D0	23	0	0	0	0	0	23
	D1	1	17	0	0	0	0	18
	D2	0	1	28	1	0	0	30
	D3	1	2	1	4	1	0	9
	D4	0	1	0	0	5	2	8
	D5	0	0	0	2	0	9	11
Totais		25	21	29	7	6	11	99

A execução do modelo com alteração para 500 épocas necessitou de quase 3 horas para treinar (02:46:18) resultando em valores mais animadores, apresentando *loss* de 0,32 e acurácia de 93,26% na época 298, porém esse tempo elevado dificultou a execução de novos testes, levando a migração de plataforma.

4.2.2 Modelo final

A partir da análise dos resultados obtidos com o modelo inicial da segunda abordagem optou-se por algumas modificações. A primeira delas foi aumentar significativamente a quantidade de épocas (500). Essa alteração elevou a acurácia e

diminuiu o *loss*, porém aumentou em muito o tempo de treinamento do modelo na plataforma PC, passando de poucos minutos para algumas horas e motivando a busca da plataforma CoLAB, equipada com GPUs e desenvolvida especificamente para esse tipo de aplicação.

Outras modificações significativas foram a opção por não utilizar *dropout* e a utilização do recurso *image augmentation* para aumentar o conjunto de dados. Essas decisões foram baseadas em exaustiva experimentação, pois a literatura e os exemplos de outras pesquisas raramente se adequam a problemas distintos.

Foram executados vários experimentos com o objetivo de possibilitar análises mais detalhadas dos resultados do modelo utilizado. A Tabela 23 apresenta os resultados dos 5 melhores experimentos.

Tabela 23 - Melhores resultados do modelo final, para o conjunto de teste

Experimento	Tempo	Época	Parâmetro	Treino	Teste
1	00:10:10	322	Loss	0,3906	0,3031
		254	Acurácia	91,03%	94,65%
2	00:10:14	370	Loss	0,3257	0,2675
		134	Acurácia	90,56%	96,88%
3	00:10:11	446	Loss	0,3391	0,2914
		126	Acurácia	89,15%	93,89%
4	00:10:16	330	Loss	0,3836	0,3405
		418	Acurácia	88,20%	93,12%
5	00:10:10	346	Loss	0,3915	0,2964
		138	Acurácia	87,26%	92,36%

O Experimento 2 foi o que apresentou os melhores resultados para *loss*, tanto no conjunto de treino quanto no teste. Na época 370 desse experimento, apesar da acurácia não ter sido a melhor (com 0,95419 contra 0,96883 na época 134), foram obtidos os melhores valores de *loss* nos dois conjuntos, treino e teste. Assim, esse modelo foi reservado para aplicação na classificação inicial das imagens completas de tábua de madeira, na terceira abordagem.

O tempo total de treinamento foi de pouco mais de dez minutos para todos os experimentos do melhor modelo. A acurácia começa estabilizar próximo da época 130 na maioria dos experimentos com bons resultados, sem melhorias significativas a partir desse ponto (Figura 39). Já o *loss* apresentou uma melhora significativa por volta da época 300 chegando ao melhor valor na época 370. Como o *loss*, no contexto de

reconhecimento de imagens, é um indicador mais relevante que a acurácia, aguarda-se até uma época em que estabilize no valor mais baixo possível (Figuras 40).

Figura 39 - Evolução da acurácia no experimento 2

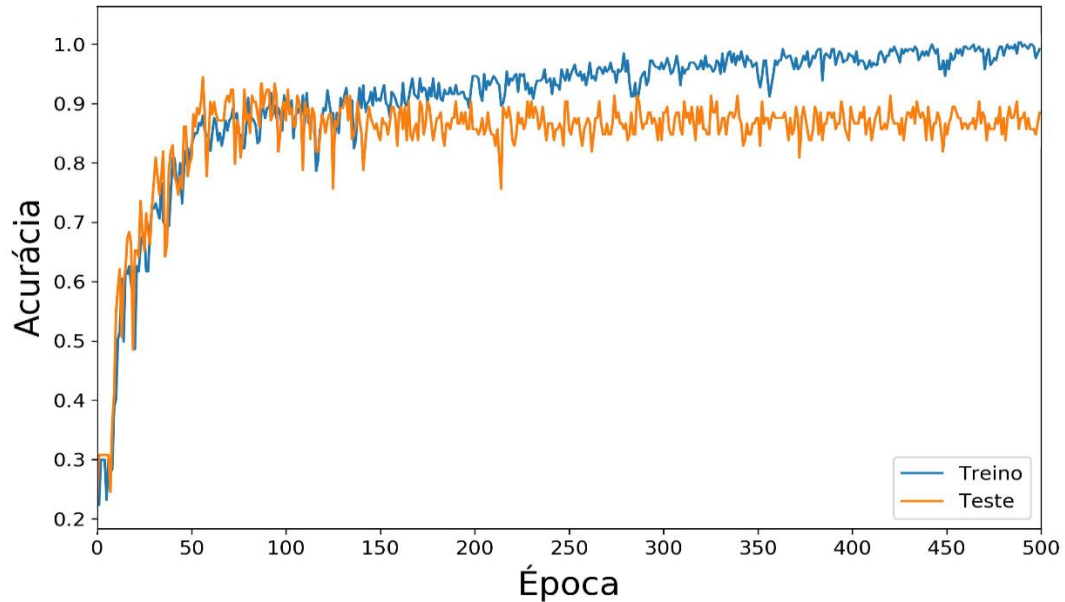
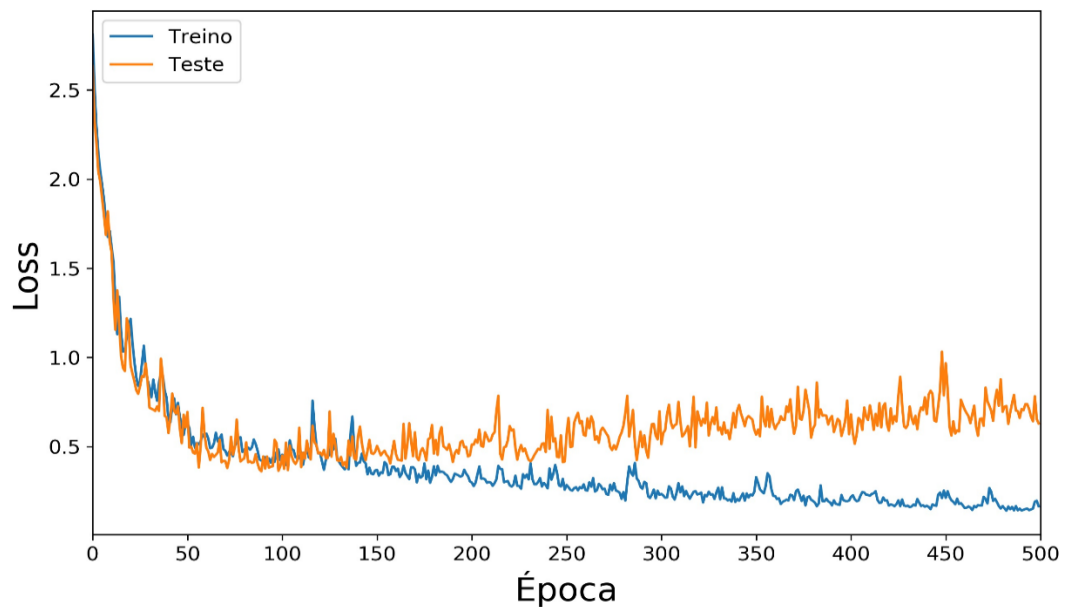


Figura 40 - Evolução do *loss* no experimento 2



Outro procedimento adotado na busca de melhorias do modelo foi, a partir da matriz de saída, analisar os valores normalizados que indicam a probabilidade de uma imagem do conjunto de testes pertencer a cada uma das classes determinadas no problema.

Quando os dados na matriz de saída indicam que a imagem possa pertencer a mais de uma classe, com probabilidades muito próximas identificam-se as imagens

que geram maior dificuldades para o modelo. Na Figura 41 observa-se que os valores circundados em vermelho indicam imagens com probabilidades em torno de 50% de pertencer a duas classes diferentes, enquanto os valores ressaltados com retângulo em azul, indicam probabilidades acima de 97% para uma classe específica, não deixando dúvidas sobre a classificação da imagem.

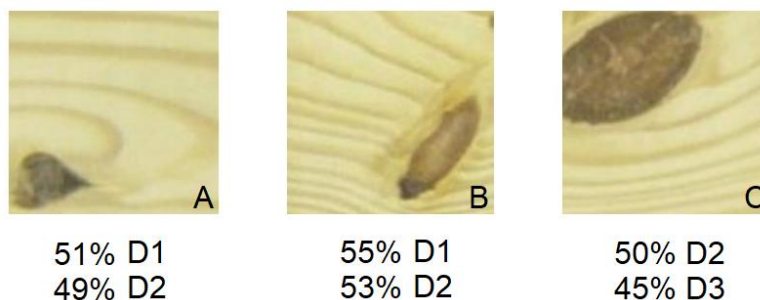
Figura 41 - Valores de saída do modelo, detalhe dos valores próximos

jupyter Modelo_DL_Convolucional_128 Last Checkpoint: 23/02/2019

	D0	D1	D2	D3	D4	D5	
	1.77	12.41	85.33	0.24	0.23	0.01	
	1.11	51.87	49.05	0.01	0.00	0.00	D1 ou D2
	1.26	92.81	5.93	0.00	0.00	0.00	
	0.04	0.03	99.89	0.04	0.00	0.00	D1 ou D2
	0.78	55.94	53.28	0.00	0.00	0.00	D2 ou D3
	0.00	0.00	45.37	50.58	0.05	0.00	
	0.04	0.00	86.99	12.36	0.62	0.00	
	0.00	0.00	0.00	0.03	33.49	66.47	D2 certamente
	0.20	0.72	98.84	0.22	0.02	0.00	
	0.33	2.24	97.07	0.32	0.03	0.00	

Buscando essas imagens no *dataset* e verificando sua classificação visual feita previamente, constata-se que apesar da semelhança das imagens e da proximidade nos valores de saída na classificação, o modelo emitiu predições em concordância com a classificação visual os três casos de proximidade do exemplo (Figuras 42 A classe real D1, Figuras 42 B classe real D1, Figuras 42 C classe real D2).

Figura 42 - Valores de predição muito próximos



Mesmo apresentando dificuldades com relação a algumas das classes, pode-se verificar na matriz de confusão que o modelo final classifica com incidência de erro muito próximo do ideal (Tabela 24).

Tabela 24 - Matriz de confusão da classificação 128x128, pelo modelo final

		Classificação Automatizada						
		D0	D1	D2	D3	D4	D5	
Visão humana	D0	23	0	0	0	0	0	23
	D1	0	18	0	0	0	0	18
	D2	0	0	29	1	0	0	30
	D3	0	0	0	9	0	0	9
	D4	0	0	0	2	6	0	8
	D5	0	0	0	0	0	11	11
		23	18	29	12	6	11	99

Para se conhecer o desempenho do modelo no equipamento de computador pessoal, efetuou-se a experimentação do modelo final também no PC.

Todos os ciclos completos de treinamento realizados no PC levaram pouco mais de 2,5 horas, gerando resultados semelhantes aos experimentos rodados no CoLAB, com uma melhoria importante na estabilidade, todos os experimentos geraram resultados muito parecidos. Essa diferença na estabilidade do modelo se deve valor atribuído a variável *seed*. No caso do PC pode ser sempre o mesmo valor, enquanto no CoLAB, a cada ciclo de treinamento, é gerada uma *seed* com valor distinto.

Foram executados seis experimentos no PC, sendo que em todos os resultados foram muito próximos. No melhor dos casos, os valores apresentados foram menor *loss* de 0,29337 na época 414 e maior acurácia na época 222 e valor de 94,65%, todos no conjunto de teste (Figuras 43 e 44, Tabela 25).

Apesar dos valores de acurácia e *loss* no PC e no CoLAB serem próximos, a experimentação no PC é mais custosa por conta do tempo demandado, inviabilizando a execução de sucessivas modificações no modelo, já que após cada modificação, por menor que seja, deve-se rodar o modelo novamente para verificar os resultados e compará-los com os resultados de modelos anteriores. Provavelmente, os bons resultados do modelo final não teriam sido alcançados, se não houvesse a possibilidade de rodar esses modelos em uma plataforma como a do CoLAB, pois não seriam experimentadas todas as centenas de modificações e efetuados os ajustes que foram necessárias para se chegar a eles.

Figura 43 - Evolução do *loss* no melhor dos experimentos rodados no PC

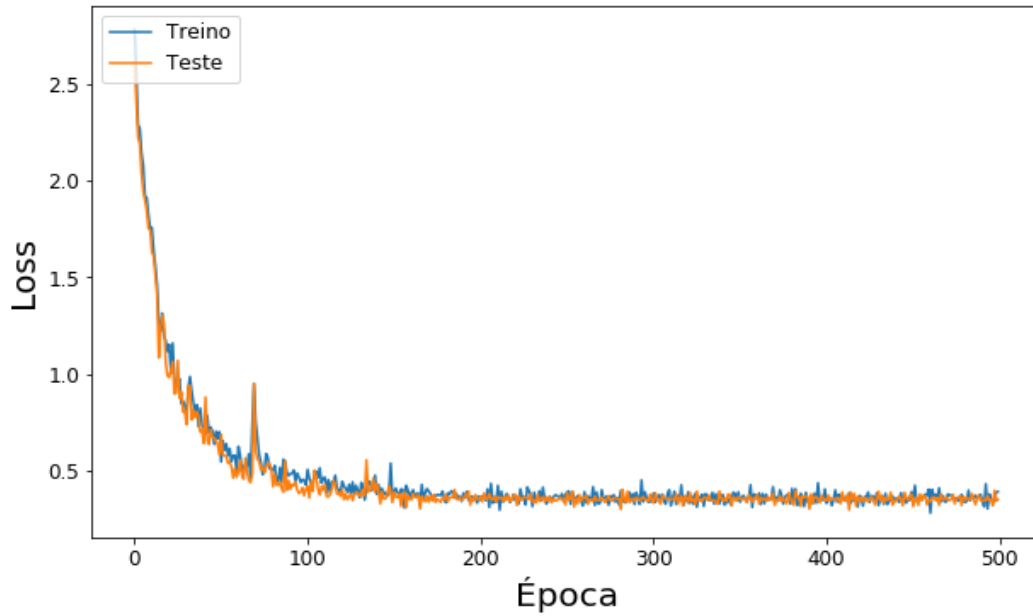


Figura 44 - Evolução da acurácia no melhor dos experimentos rodados no PC

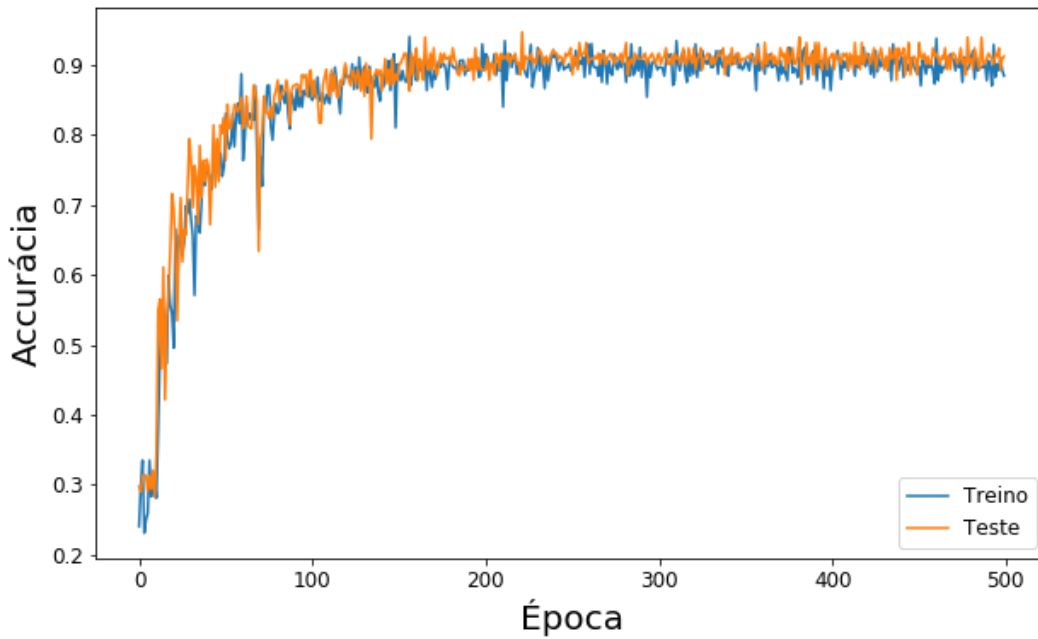


Tabela 25 - Melhor resultado dos modelos rodados no PC

Tempo	Época	Parâmetro	Resultado
	414	<i>Loss</i>	0,2933
02:36:52	222	Acurácia	94,65%

Pode-se considerar que o resultado do Modelo Final desta abordagem é satisfatório se comparado com resultados de outros trabalhos como de Almeida (2014), que aplicou técnicas como SVM e RNA e alcançou valores próximos em conjunto semelhante de imagens e classes (Tabela 26).

Tabela 26 - Resultados de trabalhos na mesma área

Autor	Técnica	Imagens (dimensão)	Classe	Resultado
Almeida (2014)	RNA (MLP)	300	6	94,79%
	SVM	(32x32)		96,88%
Autor	CNN	343 (128x128)	6	96,93%

4.3 Terceira abordagem

4.3.1 Modelo Adaptado

Efetuuou-se ajustes no modelo da abordagem anterior apenas nos formatos de entrada e saída de dados. Os valores apresentados foram abaixo do ideal para acurácia e *loss*, porém melhores do que o esperado (Tabelas 27 e 28). A partir deste aplicou-se 1500 épocas de treinamento para todos os modelos desta abordagem.

Tabela 27 - Melhor *loss* do modelo Adaptado

Época	Parâmetro	Treino	Teste
1307	<i>Loss</i>	0,4507	0,3606
	Acurácia	81,75%	89,58%

Tabela 28 - Melhor acurácia do modelo Adaptado

Época	Parâmetro	Treino	Teste
1163	Acurácia	81,08%	91,65%
	<i>Loss</i>	0,4493	0,4558

Percebe-se um comportamento da acurácia que sugere evolução se o treinamento continuasse em mais épocas, o mesmo não ocorre com o *loss* (Figuras 45 e 46). A acurácia apresenta-se em bom patamar, porém o valor do *loss* é alto e a matriz de confusão com valores muito dispersos (Tabela 29).

Figura 45 - Comportamento do *loss* - modelo Adaptado

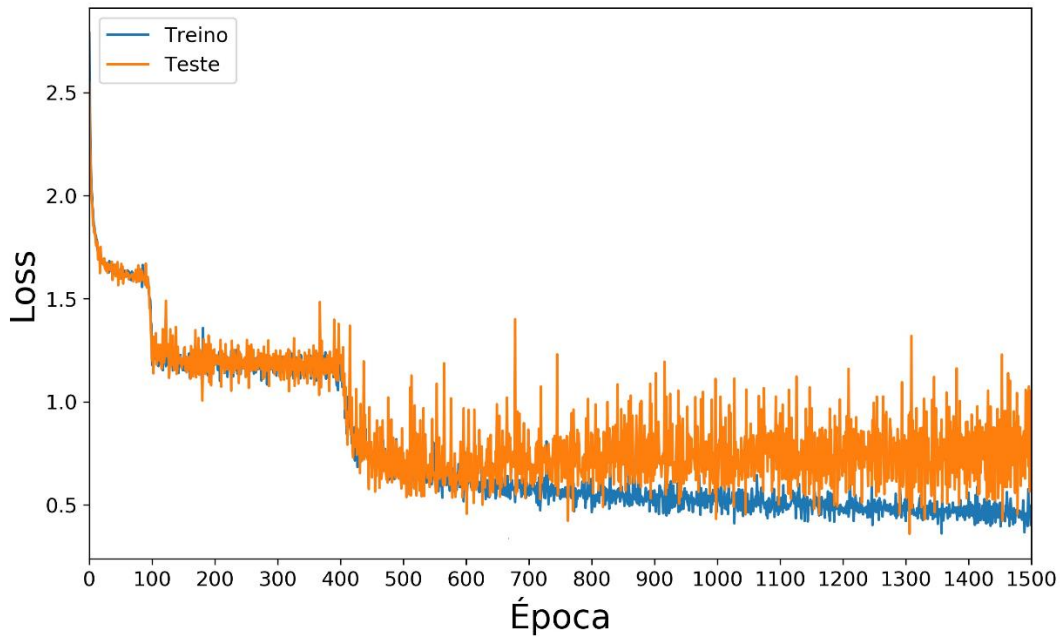


Figura 46 - Comportamento da acurácia - modelo Adaptado

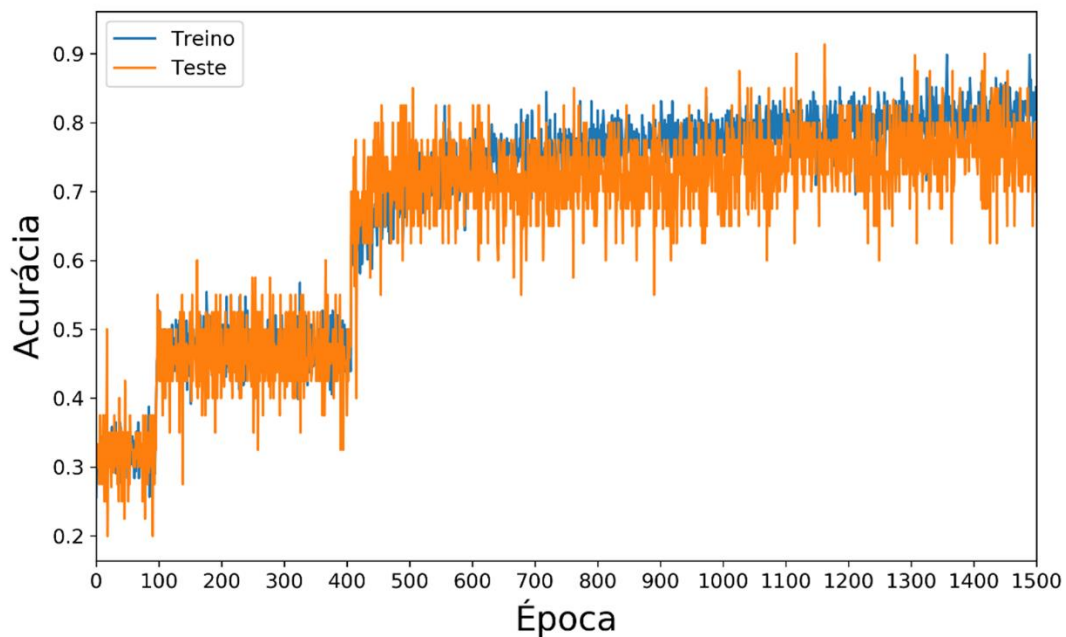


Tabela 29 - Matriz de confusão modelo Adaptado

		Classificação Automatizada				
		Super	Extra	Primeira	Segunda	Terceira
Classificação Manual	Super	10	0	0	0	0
	Extra	0	8	2	0	0
	Primeira	1	1	15	1	0
	Segunda	0	0	1	9	0
	Terceira	0	0	0	0	8

Motivado pela diferença nas quantidades de imagens de tábuas de primeira e terceira classe e também pela perceptível dificuldade entre tábuas das classes extra e primeira, efetuou-se o treinamento do modelo com *dataset* balanceado gerado por Almeida (2014), contendo 200 imagens sendo 50 para cada classe (super, extra, primeira e segunda), para verificar o impacto do desbalanceamento nos resultados (Tabela 30).

Tabela 30 - Melhores resultados do modelo Adaptado, 200 imagens balanceadas

Época	Parâmetro	Treino	Teste
255	<i>Loss</i>	0,7735	0,9333
765	Acurácia	96,87%	82,50%

Os resultados melhoraram ao trabalhar-se com 284 imagens (Tabelas 31). Em contrapartida Almeida (2014), no experimento com SVM mostra que o balanceamento do conjunto de dados pode contribuir para eficiência do modelo, apresentando 96,88% em dados balanceados contra 84,62%.

Tabela 31 - Comparativo do modelo Adaptado

Imagens	Balanceado	<i>loss</i>	Acurácia
200	sim	0.9333	82,50%
284	não	0,3606	91,65%

4.3.2 Modelo Ajustado

O modelo Ajustado de melhor performance treinou em 1500 épocas, empregou-se, assim como no modelo anterior, o algoritmo de otimização Adam, com taxa de aprendizagem de 0,0005 e de decaimento de 0,0009, definiu-se o valor 16 para o tamanho dos minilotes, foram gerados e treinados mais de sete milhões de parâmetros (7.195.460). O modelo apresentou bons resultados (Tabela 32 e 33).

Tabela 32 - Melhor *loss* do modelo aplicando *image_augmentation*

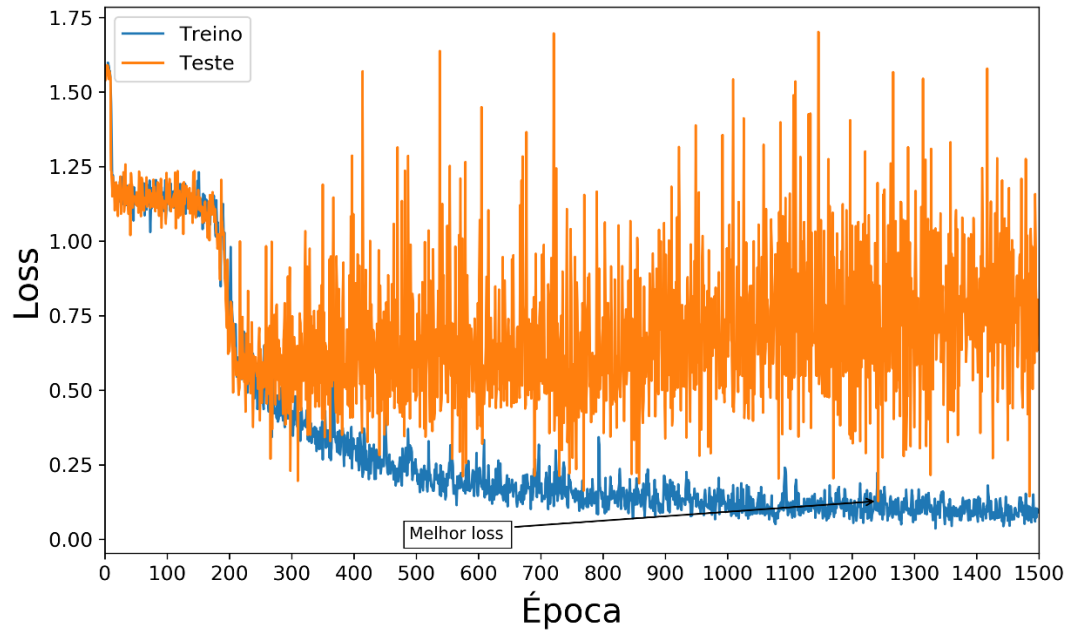
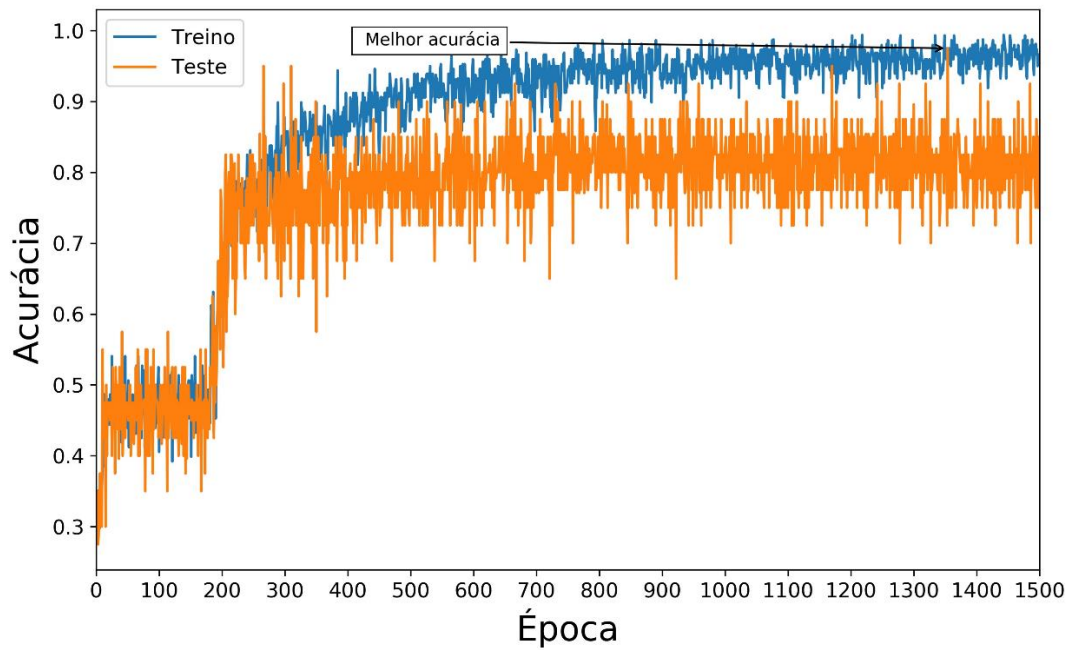
Época	Parâmetro	Treino	Teste
1243	<i>Loss</i>	0,0948	0,1684
	Acurácia	96,87%	92,50%

Tabela 33 - Melhor acurácia do modelo aplicando *image_augmentation*

Época	Parâmetro	Treino	Teste
1355	Acurácia	95,94%	94,95%
	<i>Loss</i>	0,0766	0,3265

O comportamento peculiar dos modelos a partir desta etapa se deve a utilização de alguns recursos como *batch normalization* e *global average pooling* - GAP, que efetuam a cada época operações mais drásticas para ajustes dos parâmetros treináveis, sendo que o GAP gera milhares de parâmetros não treináveis que também interferem nesse comportamento (Figuras 47 e 48).

Apesar do modelo apresentar bons resultados, observou-se na matriz de confusão uma dispersão maior que o comum entre as classes, levando ao questionamento sobre o uso da técnica aplicada por meio da função *custom_augmentation* (Tabela 34). Concluiu-se que a limitação não se deve ao modelo pois, dentre todas as variações de modelos Ajustados, esse foi o que obteve melhor performance, mesmo antes de aumentar o conjunto com as novas imagens, sendo que a melhor dessas variações chegou a 95% de acurácia.

Figura 47 - Comportamento do *loss* - modelo Ajustado**Figura 48 - Comportamento da acurácia - modelo Ajustado****Tabela 34 - Matriz de confusão modelo Ajustado**

		Classificação Automatizada				
		Super	Extra	Primeira	Segunda	Terceira
Classificação Manual	Super	10	0	0	0	0
	Extra	0	9	1	0	0
	Primeira	0	1	16	1	0
	Segunda	0	0	0	10	0
	Terceira	0	0	0	0	8

Constatou-se que ao aplicar-se o *dropout* após o GAP reduz a quantidade de parâmetros, minimizando a probabilidade de *overfitting*, porém correndo-se o risco do descarte de valores importantes para o aprendizado o que, provavelmente, impediu a evolução da acurácia neste modelo.

O modelo, por mostrar bons resultados, ficou sob observação e foi constantemente revisitado e experimentado. Porém, foi superado, pois não convergiu em nenhum dos ciclos de treinamento para números conjuntos de acurácia e *loss* que pudessem efetivá-lo como melhor opção, partindo-se para outras estratégias.

Pôde-se notar que o modelo apresenta potencial significativo devido a valores de *loss* baixos em alguns experimentos (melhor em 0,16 na época 1243) e altos para acurácia (melhor em 95% na época 1355), contudo sem nunca acontecerem na mesma época.

4.3.3 Modelo Simplificado

Analisando os valores atingidos pelo modelo percebe-se que teve boa performance se comparado aos modelos anteriores, mesmo os mais complexos que levavam horas para serem treinados (Tabela 35 e 36). Os gráficos mais comportados refletem sua simplicidade (Figuras 49 e 50). Esse modelo treinou em pouco mais de 50 minutos.

Desenvolveu-se o modelo simplificado na expectativa de que se corroborasse a teoria que afirma ser possível treinar com modelos simples e poucas *features* quando as imagens apresentam poucas características que as distinguem. Avaliando visualmente as imagens das tábuas deduz-se que apresentam pouca complexidade, levando a crer que uma classe seria categorizada com poucas características. Os resultados confirmaram essa tese, apresentando *loss* muito baixo e acurácia próxima do ideal ocorrendo nas mesmas épocas, sendo esses os resultados mais satisfatórios até esse ponto da pesquisa. A matriz de confusão confirma boa estabilidade do modelo (Tabela 37).

Tabela 35 - Melhores resultados para *loss*, modelo simplificado

Época	Parâmetro	Treino	Teste
743	<i>Loss</i>	0,5042	0,1839
	Acurácia	78,38%	94,60%

Tabela 36 - Melhores resultados para acurácia, modelo simplificado

Época	Parâmetro	Treino	Teste
703	Acurácia	78,75%	96,40%
	Loss	0,4412	0,2261

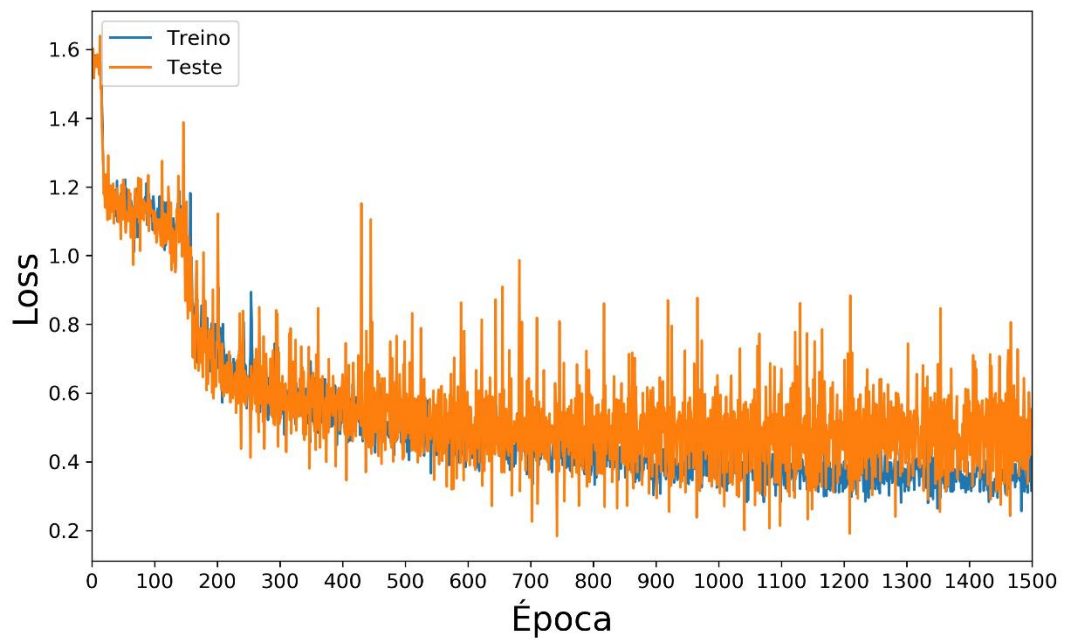
Figura 49 - Comportamento do *loss* - modelo Simplificado

Figura 50 - Comportamento da acurácia - modelo Simplificado

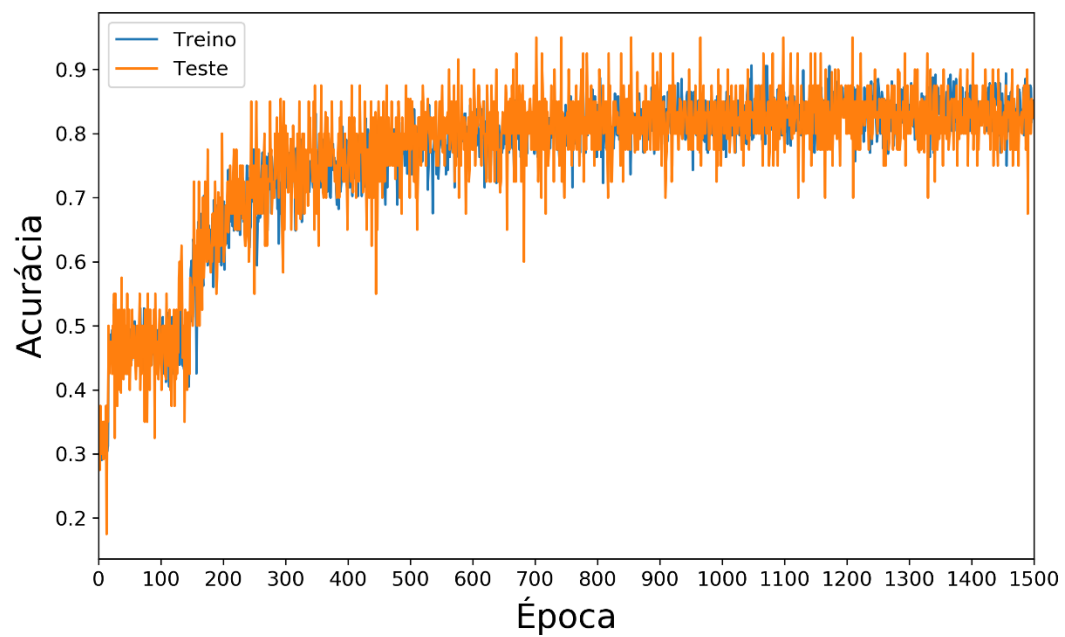


Tabela 37 - Matriz de confusão, modelo Simplificado

		Classificação Automatizada				
		Super	Extra	Primeira	Segunda	Terceira
Classificação Manual	Super	10	0	0	0	0
	Extra	0	9	1	0	0
	Primeira	0	1	17	0	0
	Segunda	0	0	0	10	0
	Terceira	0	0	0	0	8

Apesar de acurácia 96% acompanhada por *loss* 0,18 ser o melhor resultado entre todos já apresentados até aqui, incluindo os citados na maioria dos trabalhos, deu-se seguimento a um novo modelo buscando aumento na acurácia.

4.3.4 Modelo Final

Com este modelo atingiram-se, por fim, valores aceitáveis e muito próximo do considerado ideal para o tipo de problema em questão, vindo de encontro aos objetivos da pesquisa (Tabelas 38 e 39).

Os gráficos atingem picos em épocas próximas do fim do treinamento, dando indícios de que com um número mais elevado de épocas, possa-se melhorar os resultados, porém em ML esses indícios geralmente não se confirmam, mesmo porque os gráficos são gerados para representar comportamento do modelo e não tendência (Figuras 51 e 52). Por esse motivo não se criou condições para que fossem realizados mais teste, além de que os resultados apresentados já se apresentaram muito próximos do que se buscava.

Um fator que certamente contribuiria para que o modelo convergisse para melhores resultados seria o aumento substancial do conjunto de imagens, principalmente por conta das classes extra e primeira, que foram as classes que mais geraram predições incorretas, verificas em todos os modelos deste trabalho e também na matriz de confusão da classificação desta abordagem (Tabela 40).

Tabela 38 - Melhores resultados do modelo para *loss*

Época	Parâmetro	Treino	Teste
1151	<i>Loss</i>	0,2839	0,1731
	Acurácia	91,89%	97,50%

Tabela 39 - Melhores resultados do modelo para acurácia

Época	Parâmetro	Treino	Teste
1143	Loss	91,21%	97,56%
	Acurácia	0,3206	0,2078

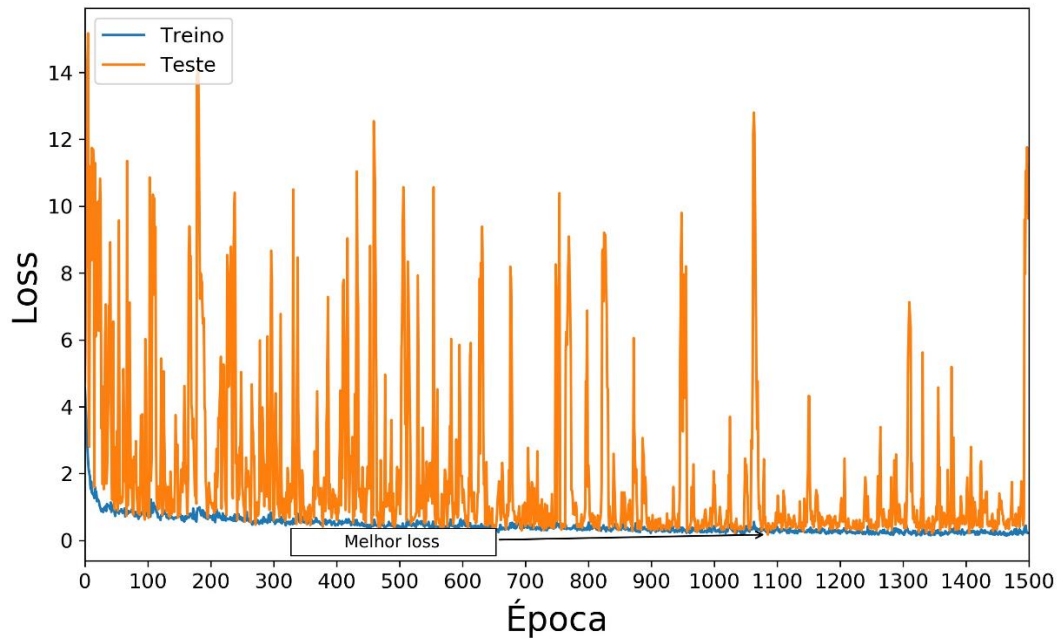
Figura 51 - Comportamento do *loss* - modelo Final

Figura 52 - Comportamento da acurácia - modelo Final

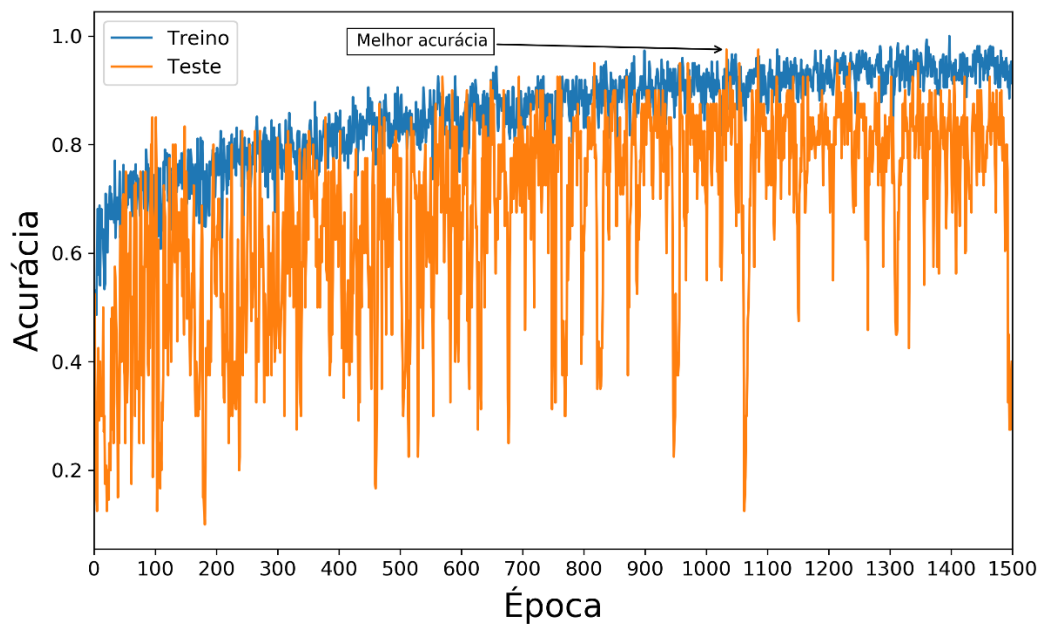


Tabela 40 - Matriz de confusão, modelo Final

		Classificação Automatizada				
		Super	Extra	Primeira	Segunda	Terceira
Classificação Manual	Super	10	0	0	0	0
	Extra	0	10	0	0	0
	Primeira	0	1	17	0	0
	Segunda	0	0	0	10	0
	Terceira	0	0	0	0	8

Almeida (2014) e Rall (2010) desenvolveram classificadores para conjuntos de dados semelhantes, considerando as 5 classes, aplicando técnicas distintas. Vieira (2016) aplica CNN e SVM, entre outras técnicas e considera 3 classes, porém mesmo com menos classes e maior quantidade de imagens que a aplicada neste trabalho seus resultados com CNN não foram satisfatórios, levando-o a concluir a técnica não seria adequado ao problema, em contrapartida atingiu 98,90% com SVM, reforçando a o que já foi dito sobre SVM, CNN e poucos. Contrapondo essa afirmação, atingiu-se na terceira abordagem desse trabalho 97,56% de acurácia acompanhada de *loss* 0,20785 na mesma época, ou 97,50% e 0,17 (Tabela 41).

Tabela 41 - Comparativo dos resultados de trabalhos na mesma área

Autor	Técnica	Imagens Tábuas	Classe	Resultado
Rall (2010)	Processamento de imagens	84	5	90,50%
Almeida (2014)	RNA(MLP)			81,25%
	SVM	200	5	79,69%
	C5.0			84,38%
Vieira (2016)	SVM	374	3	98,90%
	CNN			81,91%
Autor	CNN	284	5	97,56%

No trabalho de Almeida (2014) não foi contemplado imagens de tábuas de terceira classe pois trabalhou apenas com tábuas recém cortadas nas quais não incidem defeitos característicos de madeira seca como: nós cariados, soltos e vazados. Nesta pesquisa foram usadas tanto imagens de tábuas recém cortadas quanto secas.

5 CONCLUSÕES

Conclui-se que o *Deep Learning* com *Convolutional Neural Network* é uma técnica muito competitiva quando se trata de classificação de madeira serrada quanto a sua qualidade, considerando as normas da ABNT, fazendo da CNN uma alternativa viável para classificação de tábuas de madeira de Pinus. Atingiu-se 97,50% de acurácia com custo (*loss*) de 0,17 trabalhando com um *dataset* pouco animador, tanto pela quantidade quanto pela distribuição irregular das imagens.

Ao comparar-se os resultados com pesquisas já citadas, que relatam acurácia entre 81% e 98% em problemas semelhantes, pode-se concluir que o objetivo do trabalho foi atingido, pois desenvolveu-se um classificador que considera todas as classes da norma ABNT, sem adaptações à problemas pontuais e com acurácia considerável.

Considerando-se os resultados apresentados, pode-se afirmar que a utilização do aprendizado de máquina com DL e CNN para a classificação de tábuas de madeira mostrou-se eficiente, atingindo valores que confirmam a técnica, corroborando a hipótese levantada de que o CNN seria a técnica mais adequada.

Para produzir os resultados alcançados nesta pesquisa, foram desenvolvidas muitas variações de arquiteturas de redes neurais artificiais com aprendizado profundo aplicando CNN, partindo da classificação de blocos de imagens de 32x32 pixels e 2 classes para imagens de tábuas completas de 1000x100 e 6 classes.

Pode-se afirmar que devido ao caráter empírico das técnicas empregadas, não se deve descartar testes exaustivos com diferentes combinações das camadas da arquitetura do modelo CNN. Essa estratégia norteou a busca por um modelo eficiente e contribuiu para que se produzisse um sistema de classificação com bons resultados.

Desenvolver os modelos da primeira e segunda abordagem com panoramas mais simples, possibilitando ajustar os hiperparâmetros centenas de vezes e experimentá-los obtendo respostas de forma relativamente rápida, contribuiu significativamente para o desenvolvimento do trabalho pois permitiu a observação e análise de fatores muito importantes que foram considerados na última etapa, mais complexa e que não permitiria tal flexibilidade.

Alguns complicadores foram superados, trazendo conclusões importantes. A pouca quantidade de imagens, o desbalanceamento do *dataset*, além de pesquisas recentes que apontam para a inadequação do DL na classificação de tábuas de

madeira, foram alguns. Com relação ao desbalanceamento do conjunto de dados pode-se concluir que, em alguns casos, a redução de imagens é mais prejudicial do que a distribuição irregular. Antes do descarte de imagens para balancear o *dataset*, promoveu-se o experimento do modelo com os dois panoramas resultando em melhores índices para o conjunto com mais imagens, mesmo desbalanceadas.

Outros fatores que também poderiam atrapalhar o aprendizado, porém não impediram a convergência dos modelos foram o uso de variedades diferentes de *Pinus*, *elliottii* e *taeda*, madeira verde ou seca e tábuas aplainadas ou não.

Acredita-se que os resultados não foram melhores devido a quantidade de amostras que, apesar de não proibitiva, foi limitante. Propõe-se, em trabalhos futuros, o uso de um *dataset* de pelo menos 1000 imagens classificadas por visão humana e balanceadas, pois, mesmo que os resultados aqui apresentados não tenham sofrido com o desbalanceamento, muitas pesquisas apontam para esse fato.

Possivelmente, com alguns ajustes o modelo final aqui desenvolvido se adapte a outros panoramas e atinja bons resultados, possibilitando sua aplicação a diferentes realidades em serrarias comerciais. Pode-se, futuramente, formar um *dataset* robusto de imagens de outra espécie de árvore, para execução de experimentos e verificação de desempenho.

Com isso, foram gerados os dados finais do projeto, permitindo a análise e validação das técnicas utilizadas para a classificação das tábuas de madeira, além da comparação dos resultados com outros trabalhos que aplicaram técnicas e metodologias distintas. Tem-se, por fim, um classificador automatizado com boa acurácia, que pode ser aplicado a outros panoramas.

REFERÊNCIAS

- ABADI, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. **Arxiv**, p.1-19, 16 mar. 2016. Eprint arXiv:1603.04467. Disponível em: <<https://arxiv.org/pdf/1603.04467v2.pdf>>. Acesso em: 19 nov. 2016.
- ABIMCI-ASSOCIAÇÃO BRASILEIRA DA INDÚSTRIA DE MADEIRA PROCESSADA MECANICAMENTE. **Estudo setorial 2016**: ano base 2015. ABIMCI, 2016.
- ABNT-ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR- 11700**: Madeira serrada de coníferas provenientes de reflorestamento para uso geral: classificação. Rio de Janeiro, 1991a.
- ABNT-ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR- 12297**: Madeira serrada de coníferas provenientes de reflorestamento para uso geral: Medição e quantificação de defeitos. Rio de Janeiro, 1991b.
- ABNT-ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR- 12498**: Madeira serrada de coníferas provenientes de reflorestamento, para uso geral - Requisitos. Rio de Janeiro, 2017.
- ABPM-ASSOCIAÇÃO BRASILEIRA DE PRODUTORES DE MADEIRAS. **Pinus**: catálogo de normas de madeira serrada. Curitiba, 2001. 34 p.
- ALMEIDA, O. C. P. Classificação de Tábuas de Madeira Usando Processamento de Imagens Digitais e Aprendizado de Máquina. 105f. **Tese** (Doutorado), Faculdade de Ciências Agrônômicas de Botucatu – FCA-UNESP, 2014.
- AL-RFOU, R.; ALAIN, G.; ALMAHAIRI, A.; ANGERMUELLER, C. Theano: A Python framework for fast computation of mathematical expressions. **Arxiv**, p.1-19, 9 maio 2016. ArXiv:1605.02688v1. Disponível em: <<https://arxiv.org/pdf/1605.02688.pdf>>. Acesso em: 20 nov. 2017.
- AREL, I.; ROSE, D. C.; KARNOWSKI, T. P. **Deep Machine Learning** - A New Frontier in Artificial Intelligence Research. IEEE Computational Intelligence Magazine, 2010.
- BERGSTRA, J.; BARDENET, R.; BENGIO, Y.; KÉGL, B. Algorithms for Hyper Parameter Optimization: **Advances in Neural Information Processing Systems 24 (NIPS'2011)**. 9 p, 2011. Disponível em: <<https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization>>. Acessado em: 03 jul. 2018.
- BROWNLEE, J. **Display Deep Learning Model Training History in Keras**. 2016a. Disponível em: <<http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/>>. Acesso em: 24 nov. 2016.
- BROWNLEE, J. **Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras**. 2016b. Disponível em: <<http://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/>>. Acesso em: 24 jan. 2018.

CHOLLET, F. **Keras: Deep Learning library for Theano and TensorFlow**. 2015. Disponível em: <<https://github.com/fchollet/keras>>. Acesso em: 20 nov. 2017.

COLAB GOOGLE INC. **Colaboratory - Plataforma de desenvolvimento**. Disponível em: <colab.research.google.com>. Acessado em: 11 mar. 2019.

DENG, L.; YU, D. Introduction: Definitions and Background. In: DENG, L.; YU, D. **Deep Learning Methods and Applications**. Boston: Now Publishers, 2014. Cap. 1. p. 199-200. (7).

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning** (Adaptive Computation and Machine Learning series). MIT Press. p. 180–184, 2016. Disponível em: <<https://www.deeplearningbook.org/contents/mlp.html>>. Acessado em: 21 nov. 2018.

GOMES, J. G. et al. **Desenvolvimento e avaliação de um protótipo classificador de tábuas usando técnicas de visão artificial**. Revista Árvore, Viçosa, v. 32, n. 5, p. 949-959, 2008.

GOMES, R. C. Desenvolvimento de uma Base de Dados de Imagens Digitais De Madeira Serrada de Coníferas. 93f. **Dissertação** (Mestrado), Faculdade de Ciências Agronômicas de Botucatu – FCA UNESP, 2013.

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

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. New York: Willey & Sons, 2001.

HEBB, D. O. **The Organization of Behavior: a Neuropsychological Theory**. John Wiley & Sons, Inc. New York, 1949.

HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV R. R. **Improving neural networks by preventing co-adaptation of feature detectors**. arXiv preprint arXiv:1207.0580, 2012.

HUNTER, J. D. Matplotlib: A 2D Graphics Environment. **Computing In Science & Engineering**, [s.l.], v. 9, n. 3, p.90-95, 18 jun. 2007. Institute of Electrical and Electronics Engineers (IEEE).. Disponível em: <<http://dx.doi.org/10.1109/MCSE.2007.55>>. Acesso em: 20 mar. 2018.

IBÁ – Indústria Brasileira da Árvore. **Publicação do Relatório do Desempenho do Setor 2016**. Disponível em: <http://iba.org/images/shared/iba_2017.pdf>. Acessado em: 4 dez. 2017.

IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: **Proceedings of the 32nd International Conference on International Conference on Machine Learning**. v. 37. p. 448-456,

2015. Disponível em: <<http://proceedings.mlr.press/v37/ioffe15.pdf>>. Acessado em: 10 fev. 2019.

KAMAL, K.; QAYYUM, R.; MATHAVAN, S.; ZAFAR, T. **Wood defects classification using laws texture energy measures and supervised learning approach**. Advanced Engineering Informatics. Elsevier, October 2017. Disponível em: <<https://kundoc.com/pdf-wood-defects-classification-using-laws-texture-energy-measures-and-supervised-le.html>>. Acessado em: dez. 2018.

KANTA, G.; SANGWAN, K. S. Predictive Modeling for Power Consumption in Machining using Artificial Intelligence Techniques. 12th Global Conference on Sustainable Manufacturing. **Procedia CIRP** 26, pp 403 – 407, 2015. Disponível em: <http://ac.elscdn.com/S2212827114008853/1-s2.0-S2212827114008853-main.pdf?_tid=aa190b2afeef-11e4-bb51-0000aabb0f6b&acdnat=1432126783_a4f717dd1928cb798041843b621c2dd8> Acessado em: 12 mai. 2018.

KAUPPINEN, H. **Development of a color machine vision method for wood surface inspection**. 1999. 133f. Dissertation (Master in Electrical of Engineering)-University of Oulu, Oulu, 1999.

KINGMA, D. P.; LEI, J. B. **Adam: A method for Stochastic Optimization 3rd International Conference for Learning Representations**. San Diego: 2015. Disponível em: <<https://arxiv.org/pdf/1412.6980.pdf>>. Acessado em: 10 ago. 2018.

KHOURY JUNIOR, J. K.; PINTO, F. A. C.; QUEIROZ, D. M.; DELLA LUCIA, R. M.; RESENDE, R. C. Redes neurais para reconhecimento de defeitos de madeira serrada de eucalipto em imagens digitais. **Scientia Forestalis**, n. 70, p. 85-96, 2006.

KOIVO, A. J.; KIM, C. W. Automatic classification of surface defects on red oak boards. **Forest Products Journal**, Madison, v. 39, n. 9, p. 22-30, 1989.

KOVÁCS, Z. L. **Redes Neurais Artificiais: Fundamentos e Aplicações**. Ed. Collegium Cognitio, 1996.

KRIZHEVSKY, A.; SUTSKEVER, I; HINTON, G. E. AlexNet - Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25: **26th Annual Conference on Neural Information Processing Systems**, p. 1106–1114, 2012.

KRONKA, F. J. N.; BERTOLANI, F.; PONCE, R.H. **A cultura do Pinus no Brasil**. Páginas e Letras, 2005. 156 p.

LEBOW, P. K.; BRUNNER, C. C.; MARISTANY, A. G.; BUTLER, D. A. **Classification of wood surface features by spectral reflectance, Wood and Fiber Science**. v.28, n.1, p. 74-90, 1996. Disponível em: <<https://www.fpl.fs.fed.us/documnts/pdf1996/lebow96b.pdf>>. Acessado em: set. 2018.

LIN, H. W.; TEGMARK, M.; ROLNICK, D. Why does deep and cheap learning work so well? **Journal of Statistical Physics**, pages 1–25, 2017. Disponível em: <<https://www.deepdyve.com/lp/springer-journal/why-does-deep-and-cheap-learning-work-so-well-hNJj5L4WQH>> Acessado em: 10 out. 2018.

LORENA, A. C. **Investigação de estratégias para a geração de máquinas de vetores suporte multiclasses**. 2006. Tese de doutorado, Instituto de Ciências Matemáticas e de Computação- ICMC – USP, 2006.

MARCANO-CEDEÑO, A.; QUINTANILLA-DOMÍNGUEZ, J.; ANDINA, D. Wood defects classification using artificial metaplasticity neural network. In: **35th Annual Conference of IEEE Industrial Electronics IEEE**, 2009. Disponível em: <https://www.researchgate.net/publication/215268141_Wood_Defects_Classification_Using_Artificial_Metaplasticity_Neural_Network>. Acessado em: out. 2018.

MITCHELL, T.M. **Machine Learning**. New York: McGraw-Hill, 1997.

MONARD, M. C.; BARANAUSKAS, J. A.. Conceitos de aprendizado de máquina. **Sistemas Inteligentes - Fundamentos e Aplicações**, p. 89 -114. Editora Manole, 2003.

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

NIELSEN, M. A. Deep Learning. In: NIELSEN, M. A. **Neural Networks and Deep Learning.**: Determination Press, 2018. Cap. 6. Disponível em: <<http://neuralnetworksanddeeplearning.com/chap6.html>>. Acesso em: 08 jan. 2019.

NIMAJE, D. S.; TRIPATHY, D. P. **American Journal of Mining and Metallurgy:** Assessment of Fire Risk of Indian Coals Using Artificial Neural Network Techniques Vol. 3, No. 2, 2015, pp 43-53. Disponível em: <http://pubs.sciepub.com/ajmm/3/2/2/index.html#Figure1> Acessado em 05 set. 2017.

OBULESU, O.; MAHENDRA, M.; REDDY, M. T. Machine Learning Techniques and Tools: A Survey. **International Conference on Inventive Research in Computing Applications** (ICIRCA 2018). Coimbatore, India, 2018. Disponível em: <<https://ieeexplore.ieee.org/document/8597302>>. Acessado em: fev. 2019.

OLIVEIRA, L. E. S.; CAVALIN, P. R.; BRITTO Jr, A. S.; KOERICH, A. L. Inspeção Automática de Defeitos em Madeiras de Pinus Usando Visão Computacional. **Revista de Informática Técnica e Aplicada - RITA**, Porto Alegre, v. 25, n. 2, p. 203- 217, 2008.

ÖSTERBERG, P. Wood Quality and Geometry Measurements Based on Cross Section Images. **Thesis** (Doctorate Degree). Tampere University of Technology. Julkaisu 807 – Publication 807, 2009.

PONTI M. A.; COSTA, G.B.P. Tópicos em gerenciamento de dados e informações: Como Funciona o Deep Learning. **32nd Brazilian Symposium On Databases**. 2017. Minas Gerais. Disponível em: <<http://sbbd.org.br/2017/wp-content/uploads/sites/3/2017/10/topicos-em-gerenciamento-de-dados-e-informacoes-2017.pdf>> Acessado em: mai. 2018.

RALL, R. Processamento de imagens digitais para detecção e quantificação de defeitos na madeira serrada de coníferas de reflorestamento de uso não estrutural. 2010. 126 f. **Tese** (Doutorado) Universidade Estadual Paulista, São Paulo, 2010.

ROSENBLATT, M. The Perceptron: A probabilistic model for information storage and organization in the Brain. **Psychological review**, vol. 65, n.6, p. 386-408, 1958.

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. Cap. 1. p. 1-6.

SHIMIZU, J. Y. Pínus na silvicultura brasileira. **Revista da Madeira**, agosto:22-28, 2004.

SRIVASTAVA, N. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal Of Machine Learning Research**. Toronto, p. 1929-1958. 15 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**. Disponível em: <<https://www.tensorflow.org/>>. Acesso em: 2 nov. 2016.

UKIVA. **Twenty-one Financial Justifications for using Machine Vision**. UK Industrial Vision Association. Herts, UK. 2002. Available at <<http://www.ukiva.org/imageprocessing-software.html>> Accessed on Jan. 2015.

VIEIRA, F. H. A. Image processing through machine learning for wood quality classification. 2016. 105 f. **Tese** (Doutorado) Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2016.

WALT, S.; COLBERT, S. C.; VAROQUAUX, G. The NumPy Array: A Structure for Efficient Numerical Computation. **Computing In Science & Engineering**, v. 13, n. 2, p.22-30, 14 mar. 2011. DOI:10.1109/MCSE.2007.55. Disponível em: <<http://dx.doi.org/10.1109/MCSE.2011.37>>. Acesso em: 24 nov. 2016.

XIANG, L.; SHUO, C.; XIAOLIN, H.; JIAN, Y. **Understanding the Disharmony between Dropout and Batch Normalization Variance Shift**. Cornell University, 2018. Disponível em: <<https://arxiv.org/pdf/1801.05134.pdf>>. Acessado em: 03 jan. 2019.

YEGNANARAYANA, B. Introduction. In: **YEGNANARAYANA, B. Artificial Neural Networks**. New Delhi: Prentice Hall Of India Pvt. Ltd., 2009. Cap. 1. p. 1-24.

ZHOU, B.; KHOSLA, A.; LAPEDRIZA, A.; OLIVA, A.; TORRALBA, A. **Learning Deep Features for Discriminative Localization**. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. Disponível em: <http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf> Acessado em: 5 fev. 2019.