UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS AGRÁRIAS E ENGENHARIAS
DEPARTAMENTO DE CIÊNCIAS FLORESTAIS E DA MADEIRA

LETICIA ARAMUNI ALBERTO RIBEIRO

DESENVOLVIMENTO DE ALGORITMO DE *Image Classification* DE SUPERFÍCIES DE RUPTURA PROVOCADAS PELO ENSAIO DE FLEXÃO ESTÁTICA NA MADEIRA DE PINUS

JERÔNIMO MONTEIRO
ESPÍRITO SANTO
2022

LETICIA ARAMUNI ALBERTO RIBEIRO

DESENVOLVIMENTO DE ALGORITMO DE *Image Classification* DE SUPERFÍCIES DE RUPTURA PROVOCADAS PELO ENSAIO DE FLEXÃO ESTÁTICA NA MADEIRA DE PINUS

Monografia apresentada ao Departamento de Ciências Florestais e da Madeira da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Engenheira Industrial Madeireira.

JERÔNIMO MONTEIRO
ESPÍRITO SANTO
2022

LETICIA ARAMUNI ALBERTO RIBEIRO

DESENVOLVIMENTO DE ALGORITMO DE *Image Classification* DE SUPERFÍCIES DE RUPTURA PROVOCADAS PELO ENSAIO DE FLEXÃO ESTÁTICA NA MADEIRA DE PINUS

Monografia apresentada ao Departamento de Ciências Florestais e da Madeira da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Engenheiro Industrial Madeireiro.

Aprovada em 24 de agosto de 2022

COMISSÃO EXAMINADORA

Prof. Dr. Pedro Gutemberg de Alcântara Segundinho
Universidade Federal do Espírito Santo
Orientador

M.ª Eng. Leonor da Cunha Mastela
Universidade Federal do Espírito Santo
Examinador

M.e Augusto Melo Moulin Breda
Universidade Federal do Espírito Santo
Examinador

"A tecnologia ensinou uma lição à humanidade:
nada é IMPOSSÍVEL."

Lewis Mumford

AGRADECIMENTOS

A minha família que me incentivou a cursar o ensino superior, em especial a minha Tia Andreia Aramuni, que sempre foi incansável em seus incentivos ao estudo.

A minha mãe por sempre dispor de suas próprias necessidades em benefício meu e de meus irmãos.

Aos meus pais, avós e tios pelo suporte durante meus estudos.

Ao meu orientador Prof. Dr. Pedro Gutemberg de Alcântara Segundinho pelo acompanhamento, empatia e contribuição na pesquisa e elaboração deste trabalho.

Ao meu amigo José Almeida pela instrução e colaboração na elaboração deste trabalho.

Aos meus amigos, que durante a universidade sempre foram meu maior apoio, Lorrainy Oliveira, Letícia Rafaele, Lucas Mulin, Estefany Vaz, Sttéfany Lima e Mariana Carvalho.

A todos os meus amigos e colegas que me acompanharam durante a graduação, contribuindo com momentos de alegria e suporte nesses 5 anos.

As Professoras Doutoras Graziela Baptista Vidaurre e Rejane Costa Alves, e ao Dr. João Gabriel Missia da Silva pelo apoio, orientação, amizade e ensinamentos que foram um diferencial durante toda a graduação.

À esta Universidade e seu corpo docente pelos conhecimentos que me foram adquiridos durante a graduação.

A todos que, direta ou indiretamente, contribuíram para que eu possa estar concluindo este curso.

À Deus.

RESUMO

Neste trabalho foi desenvolvido um algoritmo de *image classification* utilizando redes neurais convolucionais (CNN). O algoritmo foi desenvolvido através da plataforma *Google Colaboratory* em linguagem *Python*, com aplicação das bibliotecas *Keras*, *TensorFlow* e *Pandas*. Foram utilizadas 2477 fotografias de corpos de prova de madeira de *Pinus*, dos quais 1050 destas imagens representam corpos de prova submetidos ao ensaio de flexão estática com superfície de ruptura aparente. O ensaio de flexão estática remete a resistência da madeira, e a ruptura obtida por este ensaio geralmente está condicionada ao pior defeito, sendo afetada por fatores como presença de nós, tipo de grã, densidade, dureza, anéis de crescimento, umidade etc. Os tipos de ruptura obtidos por este ensaio estão condicionados às características ou estado, como temperatura e umidade relativa do ar, cuja peça de madeira estava sujeita no momento do ensaio, apresentando tipos de ruptura frágil ou de ruptura normal, correlacionadas com a resistência da peça. Com a finalidade de desenvolver um algoritmo que possibilitasse a classificação do tipo de superfície de ruptura, segundo a norma ASTM D-143 (2007), de forma automatizada, minimizando erros que possam ocorrer em uma análise manual. Para tal, testes de qualidade de aprendizagem de máquina foram implementados, e com o auxílio de *Callbacks*, da biblioteca *Keras,* as métricas de *accuracy* (acuracidade) e *loss* (perda) foram aplicadas. A partir destas, foi estabelecido o limite do aprendizado do modelo, que atingiu a acuracidade de 58,23% e perda de 68,30%. Baseado nesses resultados, compreende-se ainda há possibilidade de crescimento para o modelo, crescimento este que pode ser atingido com o cumprimento de pelo menos uma, de duas hipóteses. A primeira hipótese supõe que o banco de dados fornecido para o algoritmo limitou a capacidade de aprendizagem, sendo necessário buscar, para trabalhos futuros, um banco de dados mais extenso e diverso quanto aos tipos de superfície de ruptura. Como segunda hipótese, compreende-se que para a realização das fotografias, um padrão de ângulo, altura, iluminação e área de visualização poderia facilitar a leitura das imagens pelo modelo, o que poderia ocasionar em um aumento da acuracidade e diminuição da perda.

Palavras-chave: *Python*, *Convolutional Neural Network*, Visão computacional.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

O mundo globalizado, com todo seu dinamismo, exige que os processos, sejam industriais ou manuais, se transformem em processos mais dinâmicos, otimizados, que apresentem os melhores resultados em menor tempo de processamento. Através da programação, os desenvolvedores encontraram formas de otimizar os processos e desenvolver novas ferramentas em todos os campos de atuação imagináveis.

A programação se trata do processo de escrita, testes e manutenção de códigos e algoritmos. A lógica de programação é a técnica de descrever pensamentos e sequências de ações, para atingir determinado objetivo. O algoritmo é uma sequência de passos finitos com o objetivo de solucionar um problema (LOPES; GARCIA, 2002).

Dessa forma, utilizando da lógica de programação, todo tipo de ação, se descrita passo por passo até sua solução, pode ser solucionada por um algoritmo. Portanto, o algoritmo não é a solução de um problema, e sim, um conjunto de passos que levam à solução de um problema “x” (LOPES; GARCIA, 2002).

Estruturas de madeira estão cada vez mais incorporadas no mercado, por suas características favoráveis ao setor, sejam estas a capacidade estética, a resistência ao fogo, as características térmicas e acústicas, ou as características físicas e mecânicas deste material. Além disso, que a madeira é um material sustentável, que serve como um depósito natural de carbono.

A madeira de floresta plantada se destaca no setor florestal e da construção civil, diante das restrições para o uso em larga escala das espécies nativas. As propriedades mecânicas da madeira possuem grande variabilidade, mesmo em elementos estruturais de espécies iguais por ser um material heterogêneo, anisotrópico e vulnerável aos agentes externos e ao processamento industrial.

As propriedades mecânicas da madeira são influenciadas por várias características, como a massa específica básica (densidade básica), estrutura anatômica, tais como as fibras e vasos, pela constituição química e por fatores como a umidade da madeira, que é inversamente proporcional a resistência mecânica desse material (KOLLMANN; CÔTÉ JUNIOR, 1968).

Com o uso de elementos estruturais de madeira serrada vinda de floresta plantada, é importante realizar a classificação visual manual e mecânica peça a peça

para determinar a classe de resistência na incidência dos defeitos de crescimento e de secagem, cujas propriedades mecânicas podem sofrer grande variabilidade entre as peças do lote, inviabilizando a classificação do lote como homogêneo, como é recomendada pela norma brasileira ABNT NBR 7190-1 (2022).

As propriedades físicas e mecânicas e a presença de defeitos na peça são parâmetros avaliados se tratando da madeira dedicada para uso estrutural, entre as principais propriedades mecânicas da madeira, estão a resistência a esforços devido a flexão, compressão, tração, cisalhamento e fendilhamento (ARAÚJO, 2002).

Dentre estas dão-se atenção à flexão, em especial ao ensaio de flexão estática, cujas propriedades são consideradas referências das classes de resistência da norma europeia ISO 16598 (2015). A ruptura, obtida por nesse ensaio, normalmente é condicionada ao pior defeito, portanto há uma relação com os defeitos e nós presentes na peça, assim como com a densidade básica, anéis de crescimento e resistência do material (SANTOS, 2019).

1.1. O problema e sua importância

A classificação visual para a análise de defeitos, nós e padrões de ruptura em peças de madeira não seguem uma técnica padrão, são realizadas de forma visual e manual, sujeitas a divergências e erros causados pelo avaliador.

Conforme o aumento do consumo de madeira e da industrialização dos processos de transformação deste material, as análises que visam avaliar as propriedades deste material e que seguem sendo aplicadas através de técnicas manuais e repetitivas, se transformam cada vez mais em gargalos que retardam os processos que englobam as transformações e processamentos da madeira.

Algoritmos que promovam a otimização de tais análises e técnicas podem melhorar o desempenho de um processo produtivo ao reduzir gargalos e aumentar a eficiência e eficácia na entrega de dados. Soluções essas que podem beneficiar diversos negócios e indivíduos, considerando o dinamismo do setor florestal.

O avanço tecnológico de *hardware* e *software* possibilitou o uso de técnicas de classificação computacional e reconhecimento de padrões visuais, os quais podem ser aplicados visando a contribuição na interpretação automática de imagens.

Considerando tais pontos, este trabalho apresentará um algoritmo voltado para análise de imagens de superfícies de madeira *Pinus*, visando proporcionar a

otimização da análise visual do tipo de superfície de ruptura provocado pelo ensaio de flexão estática.

1.2. Objetivos

1.2.1. Objetivo geral

Desenvolver um algoritmo em linguagem de programação *Python* para identificação e classificação do tipo de superfície de ruptura gerada pelo ensaio de flexão estática.

1.2.2. Objetivos específicos

- Coletar as imagens de superfícies de ruptura em madeira pinus provocadas pelo ensaio de flexão estática;
- Classificar e separar as imagens coletadas;
- Desenvolver o código do algoritmo de análise de imagens;
- Calibrar o algoritmo quanto a leitura dos tipos de superfícies de ruptura;
- Realizar testes de qualidade para verificar a acuracidade do algoritmo.

2. REVISÃO DE LITERATURA

2.1. Madeira de *Pinus*

Em 2019, a área total de árvores plantadas totalizou 9 milhões de hectares. Desse total, 18% correspondem ao gênero *Pinus*, com 1,64 milhão de hectares (IBÁ, 2020). Segundo a Sociedade Brasileira de Silvicultura (SBS, 2007), a produção de madeira serrada atingiu 23,8 milhões m³, sendo o *Pinus* responsável por mais de 38% deste total.

A espécie de coníferas do gênero pinus além de produzir madeira de excelente qualidade visual, ainda é uma candidata excelente a estudos para a determinação de propriedades físicas, químicas e mecânicas, isto porque, se trata de uma espécie de rápido crescimento, cujo desenvolvimento comercial é endossado pelas condições propicias do solo e clima do país (ROVEDA FILHO, 2006).

O gênero Pinus compreende mais de uma centena de espécies, dentre estas, as espécies mais plantadas são *P. taeda* e o *P. elliottii* (BALLONI, 2009). De acordo com Camargo (2016), a madeira de P. taeda é a mais plantada do gênero, e é indicada para a produção de celulose, construções, dormentes, laminação, fabricação de móveis e para serraria. Balloni (2009) ainda acrescenta que o *P. elliottii* é majoritariamente direcionado para o processamento mecânico e a extração de resina.

A madeira de *Pinus taeda* e *Pinus elliottii* são similares, principalmente na aparência, ambas apresentam alburno de coloração branco amarelado e o cerne marrom avermelhado. Quanto às suas propriedades, a madeira de *P. elliottii* é mais densa, resistente, dura e rígida, sendo, portanto, mais resistente ao impacto que a madeira *P. taeda* (FOREST PRODUCTS LABORATORY, 1999).

Devido à crescente demanda por madeira em tora e por madeira serrada, este gênero está sendo utilizado para suprir a demanda do setor, principalmente dos setores moveleiro e da construção civil, como ressaltado por Dobner Júnior (2012).

2.2. Flexão estática

Por meio do ensaio de flexão é possível obter informações importantes a respeito do material quando submetidos a esforços de flexão. As principais propriedades obtidas através do ensaio de flexão são:

- Módulo de ruptura na flexão;
- Módulo de elasticidade;
- Módulo de resiliência;
- Módulo de tenacidade.

A resistência à flexão estática pode ser definida como a resistência da madeira a forças ao longo do seu comprimento. Para a determinação da flexão estática uma carga é aplicada tangencialmente aos anéis de crescimento em uma amostra apoiada nos apoios (MORESCHI, 2014).

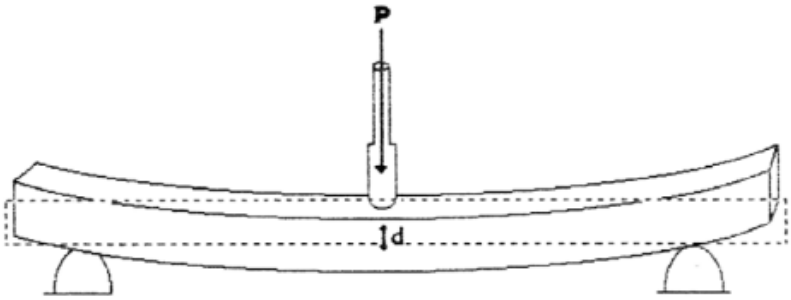

Figura 1 - Esquema de ensaio de flexão estática (P = carga; d = deformação)
Fonte: Moreschi (2014).

As normas regulamentadoras servem como instrumentos para a realização do ensaio, dentre estas estão:

- NBR 7190-1 (ABNT, 2022);
- International Organization for Standardization – ISO 13910 (2005).

2.2.1. Tipos de superfícies de ruptura

Os tipos de ruptura gerados pela flexão estática são classificados formas de superfície, subdivididos em ruptura frágil e ruptura normal. Ruptura frágil apresenta as

superfícies de tração com ruptura, compressão e cisalhamento horizontal, as peças que apresentam tais classificações apresentam menor resistência da madeira. Ruptura normal representa as classificações de tração simples, tração desviada e tração com desfibramento.

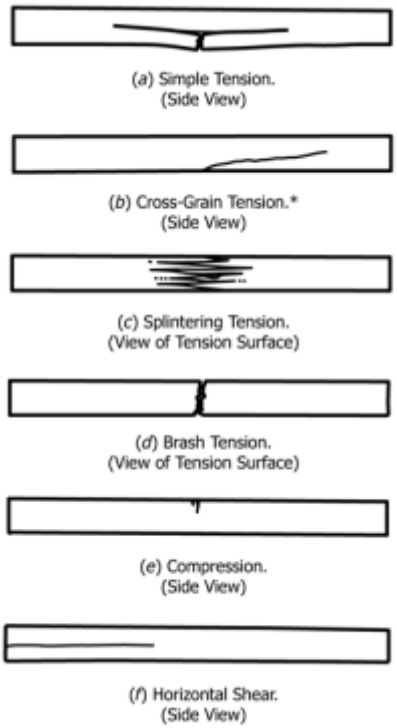

Figura 2 - Tipos de superfícies de ruptura em flexão estática, onde (a) tração simples (*Simple tension*), (b) tração desviada (*Cross-Grain tension*), (c) Tração com Desfibramento (*Splintering tension*), (d) Tração com Ruptura (*Brash tension*), (e) Compressão (Compression), e (f) Cisalhamento horizontal (*Horizontal shear*).
Fonte: ASTM D-143 (2007).

Os tipos de superfícies de ruptura da madeira apresentam uma diferença considerável, que irá depender de fatores como a dureza, resistência, da grã, dos defeitos, da secagem da peça, entre outros (CALONEGO; SEVERO; BRITO, 2013), porém, o estudo a cerca dessa falha provocada pelo ensaio de flexão estática não é muito expressivo, sendo, necessário estudos futuros para compreender o comportamento e as influência as quais estão sujeitas tais falhas.

Tomando as falhas referentes a ruptura normal, Record (2004) as descreve iniciando pela tração simples, como uma falha que ocorre quando há tração direta em duas das madeiras no lado inferior da viga devido a uma tensão de tração paralela à

fibra, (Fig. 2, item (a)) isso é comum em vigas de grã reta, principalmente em madeiras de clima temperado.

A tração desviada é causada por uma força de tração que atua transversalmente a grã (Fig. 2, item (b)). Esta é uma forma comum de falha onde a viga tem grãs irregulares em seu lado inferior (RECORD, 2004).

A tração com desfibramento consiste em um número considerável de rupturas de tração simples, produzindo uma ruptura irregular ou lascada na superfície inferior da viga (Fig. 2, item (c)). Isso é comum em madeiras duras. Neste caso, a superfície da fratura é fibrosa (RECORD, 2004).

Quando as classificações referentes a ruptura frágil, observamos a tração com ruptura que a viga falha por uma ruptura limpa (Fig. 2, item (d)) (RECORD, 2004).

A compressão (Fig. 2, item (e)) ocorre em madeiras verdes, a tensão de compressão paralela às fibras faz com que elas se deformam ou dobrem como em um teste de compressão de ponta a ponta. Frequentemente, duas ou mais falhas se desenvolvem aproximadamente ao mesmo tempo (RECORD, 2004).

Segundo Record (2004), a ruptura por cisalhamento horizontal, na qual a parte superior e inferior da viga desliza uma sobre a outra por uma fração de seu comprimento em uma ou em ambas as extremidades (Fig. 2, item (f)), é bastante comum em material seco ao ar e em material verde quando a relação entre a altura da viga e o vão é relativamente grande. Não é seguro, ao projetar grandes vigas de madeira, usar tensões de cisalhamento mais altas do que aquelas calculadas para vigas que falharam no cisalhamento horizontal. O efeito de uma falha no cisalhamento horizontal é dividir a viga em duas ou mais vigas cuja resistência combinada é muito menor que a da viga original

2.3. Linguagem de programação Python

Python é uma linguagem de programação divulgada em 1991 por Guido Van Rossum, que devido a sua sintaxe simples, fácil interpretação e grande capacidade de funcionalidades é uma das linguagens de programação mais utilizadas no mundo. (CHUN, 2006). Foi desenvolvida com o objetivo de criar uma ferramenta de programação que acelerasse o processo de desenvolvimento de softwares (LIMA, 2021).

Essa linguagem possui várias características entre as quais se destacam: a indentação, o que obriga o código a ser sempre organizado e legível; ser interpretada, o que permite a flexibilidade de mudança entre os sistemas operacionais; multiparadigma, o que possibilita o desenvolvimento em várias abordagens diferentes como orientada a objetos, imperativa, funcional e procedural (ABREU, 2016).

2.3.1. Algoritmo

Algoritmos são sequencias de instruções desenvolvidos para a solução de problemas. Segundo Manovich (2001), qualquer processo ou tarefa pode ser reduzido a um algoritmo, definindo o algoritmo como "uma sequência de operações simples que um computador pode executar para alcançar uma tarefa dada".

O constante avanço tecnológico e a inserção da inteligência artificial torna impossível visualizarmos um mundo onde os algoritmos não façam parte, pois estes estão inseridos na base de toda a sociedade contemporânea (ANDRIJIC, 2019).

2.4. Visão computacional

A visão computacional tem como objetivo extrair informação útil das imagens, para tal, tenta-se descrever o mundo que vemos em imagens e reconstruir suas propriedades como forma, iluminação e distribuição de cor em uma função bidimensional, $f(x,y)$, que pode ser lida por um computador, em que x e y representam coordenadas de um plano. A amplitude de f em qualquer par de coordenadas é chamada de intensidade da imagem (FELISBERTO, 2015).

Portanto a imagem digital é composta por pixels que são por definição um número finito de elementos, que correspondem a uma localização e valor de intensidade específico (GONZALEZ; WOODS,2008). A representação dos pixels é feita através de uma matriz de duas dimensões, em que (x,y) são coordenadas das células dessa matriz (FELISBERTO, 2015).

Inúmeras áreas são relacionadas à visão computacional, e muitas técnicas desenvolvidas por tais áreas podem ser utilizadas para recuperar informações das imagens, como processamento de imagens, computação gráfica, reconhecimento de padrões e inteligência artificial.

2.5. Redes Neurais Artificiais

O estudo sobre as redes neurais iniciou-se inspirado na maneira como o cérebro humano realiza com velocidade uma infinidade de tarefas complexas como, por exemplo o reconhecimento visual. O entendimento acerca do funcionamento, comunicação e conexões realizadas pelas redes neurais biológicas ainda é um tema que levanta muitas questões entre os cientistas, mas é comumente aceito que todas as funções neurais biológicas são armazenadas nos neurônios e nas conexões entre eles (BARCELLOS, 2011).

Compreende-se que, o processo de aprendizado é definido com a criação de novas conexões e com a modificação das conexões interligadas, gerando assim um novo padrão para a aprendizagem. Contemplando esse conceito, foi modelado um conjunto de neurônios artificiais que agrupados constituem as chamadas redes neurais artificiais (FLORENCIO, 2020)

Para constituir uma rede neural, necessita-se de um neurônio artificial, que é a unidade fundamental básica de uma rede neural artificial. Ele possui a função de ativação, que define a amplitude do sinal de saída a um valor finito. O neurônio é definido por três elementos básicos: um sinal de entrada x_j conectado ao neurônio k que é multiplicado pelo peso da sinapse w_{kj}(FLORENCIO, 2020).

Ainda segundo Florêncio (2020), o neurônio artificial pode ser descrito pelas Equações 1 e 2, representadas abaixo:

$$W_k = \sum_{j=1}^{m} W_{kj} X_j \qquad (1)$$

$$y_k = \varphi\,(u_k + b_k) \qquad (2)$$

Onde $X_1, X_2, \ldots X_m$ são os sinais de entrada, $W_{k1}, W_{k2}, \ldots W_{km}$ representam os pesos sinápticos do neurônio k e b_k correspondente ao *bias*, que é o responsável por realizar o deslocamento da função de ativação, definida por $\varphi\,(.)$ (FLORENCIO,2020).

No processo de construção de uma rede neural, os neurônios são agrupados em camadas, camadas essas que definirão a profundidade da rede neural. O modo de organização das camadas e a sua forma de aprendizado irão definir a arquitetura da rede neural artificial. As redes neurais convolucionais tem como uma das

características principais o uso de mapas de convolução como conjunto de pesos compartilhados entre os vários neurônios das camadas de convolução

2.5.1. Rede Neural Convolucional (*Convolutional Neural Network*)

A convolução se trata de uma operação matemática entre duas funções, f e g, que produz uma terceira função que será interpretada como a função modificada de f. No processamento de imagens, como a imagem é vista como uma função bidimensional, a convolução contribui para a detecção de bordas, suavização de imagem, extração de atributos e outras aplicações (ZHANGYANG, 2021)

A rede neural convolucional (*CNN – Convolutional Neural Network* ou *Convnet*) é uma classe de rede neural artificial do tipo *feed-forward*, o que significa que a saída de uma camada é usada como entrada para a próxima camada da rede neural, ou seja, não há *loops* na rede e as informações sempre serão alimentadas para a frente e nunca enviadas de volta (FLORENCIO, 2020).

Convnets são projetadas para otimizar o tempo de treinamento através de *backprooagation*, portanto, desenvolvidas para processar dados de armazenamento na forma de múltiplas matrizes de uma dimensão para sinais e sequências, incluindo as linguagens, duas dimensões para imagens e espectrogramas e três dimensões para vídeos e imagens volumétricas (MACHADO, 2020).

Portanto, a rede neural convolucional se trata de um algoritmo de *deep learning* que analisa uma imagem, absorvendo novas informações sobre a imagem a cada nova camada, sendo assim capaz de aprender de forma autônoma as características que compõe um objeto (MACHADO, 2020).

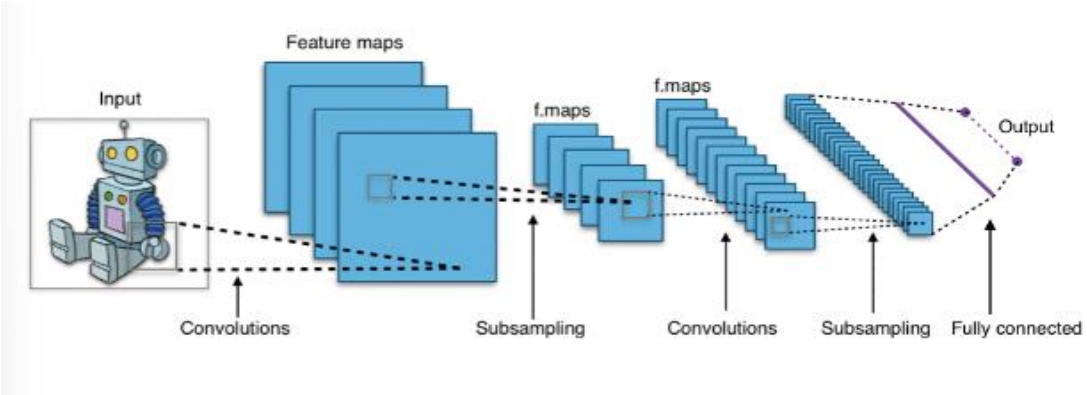

Figura 3 - Exemplo de uma estrutura da CNN
Fonte: MACHADO (2020).

No primeiro momento a CNN aplica um filtro convolucional na imagem. Esse filtro possui valores binários que quando multiplicados com as matrizes da imagem ressaltam *features* (caracteres) que o algoritmo está procurando, com o objetivo de contribuir para a visualização do modelo quanto às linhas verticais que compõem o objeto (MACHADO, 2020).

Portanto, a primeira operação convulacional visa extrair as características gerais (*low-level features*), como linhas, cor, orientação e detalhes menores do objeto da imagem. Após essa identificação, o modelo passa a buscar pelas características mais complexas (*high-level features*) do objeto, assim cada camada vai aumentando a complexidade, com mais riqueza de detalhes sendo absorvida pelo modelo (MACHADO, 2020).

Assim, o filtro convulacional é aplicado várias vezes na mesma imagem em uma rede convulacional, até que seja absorvido pelo modelo todos os *high-level* e *low-level features* do objeto em questão, para assim, como demonstrado na Figura 2, o modelo transforma todas as matrizes geradas em cada camada em um único vetor de apenas uma coluna, onde sobre ele é aplicado o *back propagation* da rede neural que através de um classificador, une todas as informações obtidas para traduzi-las na identificação do objeto (MACHADO, 2020).

2.6. Considerações da revisão de literatura

Por fim, pode-se pontuar que os tipos de ruptura provocados pelo ensaio de flexão estática na peça de madeira são um indicativo da resistência e dureza do material, porém não se sabe muito sobre as relações que influenciam cada tipo de ruptura. A partir disso, compreende-se que a utilização de técnicas de visão computacional a fim de facilitar o estudo acerca dos tipos de ruptura se torna necessário com a premissa de desenvolver novas ferramentas que possam contribuir para a realização de trabalhos que explicitem a relação das falhas com as características e estado do material.

3. MATERIAIS E MÉTODOS

A abordagem proposta neste trabalho para identificação e classificação de imagens de superfícies de ruptura geradas pelo ensaio de flexão estática é composta por quatro (04) fases. A primeira fase consistiu na captura das imagens das peças de madeira de *Pinus* com e sem superfície rompida pelo ensaio de flexão estática. A segunda fase consistiu na segmentação manual destas imagens para utilizá-las no algoritmo posteriormente. Na terceira fase, iniciou-se o desenvolvimento do algoritmo de identificação e classificação das imagens de forma automatizada. A última fase, consistiu no teste de qualidade do algoritmo.

3.1. Obtenção e preparo das imagens

O presente estudo foi realizado com as fotografias de corpos de prova do gênero Pinus, especificamente as espécies *Pinus taeda* e *Pinus elliottii*, que foram submetidos ao ensaio de flexão estática, sejam estas, fotografias da superfície de ruptura ou da madeira sem defeitos e/ou ruptura.

Os corpos de prova utilizados como banco de dados neste estudo foram fornecidos por Santos (2019) à Breda (2022), que utilizou o material para a realização do ensaio de flexão estática segundo a norma ASTM D143-94 (1995), a fim de obter as propriedades deste material, que foram utilizadas para a realização desta monografia.

Para a realização das fotografias, um local foi preparado com boa iluminação e fundo neutro (preto ou branco) para posicionar as peças. Primeiramente, as peças de madeira foram limpas, a fim de retirar a poeira e outras impurezas. Após, as peças foram separadas entre as que apresentavam algum tipo de superfície de ruptura e as que não apresentavam. Por fim, com o auxílio de um aparelho celular, as peças foram fotografadas em todos os sentidos da madeira (Radial, Tangencial e Longitudinal).

Ao todo foram obtidas 1427 imagens de superfícies limpas, sem ruptura e defeitos aparentes, e 1050 imagens de peças que representavam diferentes tipos de superfícies de ruptura provocadas pelo ensaio de flexão estática.

(a) (b)

Figura 4 - Amostra de imagens com superfície rompida (a) e sem superfície rompida (b)

Fonte: A Autora (2022).

3.2. Segmentação das imagens

Para a segmentação e classificação visual das imagens com superfície rompida seguiu-se a representação dos tipos de superfícies de ruptura em flexão estática apresentados pela norma técnica ASTM D-143 (2007), como demonstrado na Figura 2.

As imagens representantes das superfícies lisas e sem rompimento, assim como as imagens das diferentes superfícies de ruptura provocadas pelo ensaio de flexão estática foram classificadas em um processo manual e visual em pastas no *Google Drive* para utilização posterior no treinamento do algoritmo.

Tabela 1 - Banco de dados de imagens

Pasta	**Correspondência**	**Quantidade de Imagens**
Um (01)	Sem Superfície Rompida	1427
Dois (02)	Ruptura Frágil - Compressão	51
Três (03)	Ruptura Frágil - Tração com Ruptura	285
Quatro (04)	Ruptura Frágil - Cisalhamento Horizontal	103
Cinco (05)	Ruptura Normal - Tração Simples	356
Seis (06)	Ruptura Normal - Tração Desviada	128
Sete (07)	Ruptura Normal - Tração com Desfibramento	127

Fonte: A autora (2022).

A Figura 5 apresenta uma imagem referente a cada tipo de superfície de ruptura provocada pelo ensaio de flexão estática. Com a Tabela 1 é possível relacionar a representação das imagens com os tipos de superfície rompida.

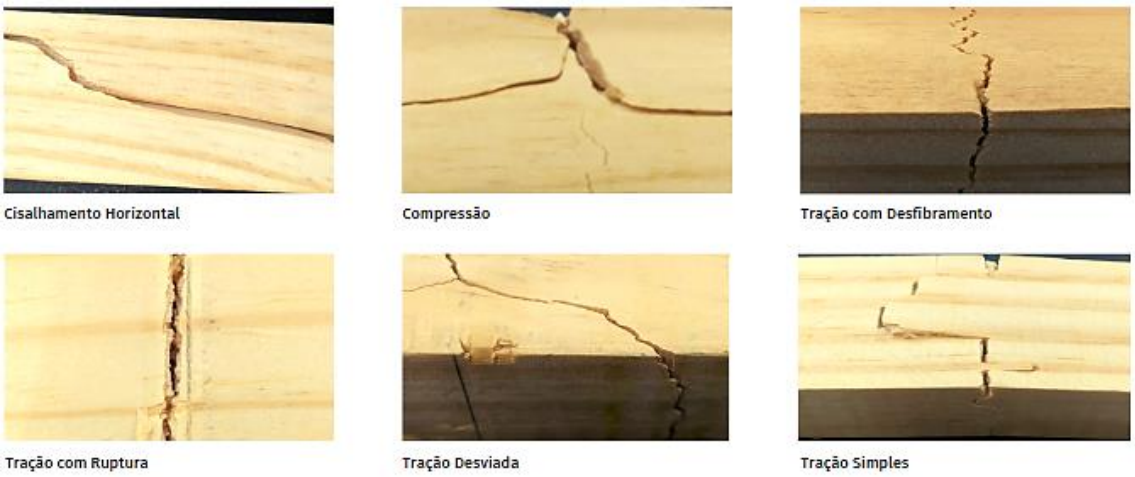

Figura 5 - Imagens representantes de cada tipo de superfície de ruptura na madeira *Pinus*

Fonte: A autora (2022).

3.3. Desenvolvimento do algoritmo

Para o desenvolvimento do algoritmo proposto, utilizou-se da linguagem de programação *Python* e de bibliotecas *Keras*, *Pandas* e *TensorFlow*. Para a escrita do código utilizou-se a plataforma do *Google Colaboratory* que permite a execução e escrita do código no navegador do *Google.*

3.3.1. *Numpy*

As funções de álgebra linear *NumPy* contam com *BLAS* e *LAPACK* para fornecer implementações eficientes de baixo nível de algoritmos de álgebra linear padrão. Essas bibliotecas podem ser fornecidas pelo próprio *NumPy* usando versões C de um subconjunto de suas implementações de referência, mas, quando possível, são preferidas bibliotecas altamente otimizadas que aproveitam a funcionalidade do processador especializado (NUMPY, 2022).

As funções de álgebra linear da biblioteca *Numpy* foram utilizadas no desenvolvimento deste algoritmo para a vetorização e transformação das imagens em linguagem binária.

3.3.2. *Pandas*

Pandas se trata de uma biblioteca de código aberto licenciada pelo BSD (*Berkeley Software Distribution*) que fornece estruturas de dados de alto desempenho e fáceis de usar e ferramentas de análise de dados para a linguagem de programação *Python* (PANDAS, 2022).

O *pandas* permite trabalhar com diferentes tipos de dados, neste trabalho o *pandas* foi aplicada para a leitura de:

- Dados ordenados;
- Matrizes;
- E funções.

3.3.3. *TensorFlow*

TensorFlow é uma biblioteca de *software* de código aberto para computação numérica de alto desempenho. Sua arquitetura flexível permite fácil implantação de computação em uma variedade de plataformas (*CPUs, GPUs, TPUs*) e de *desktops* a *clusters* de servidores a dispositivos móveis e de borda (TENSORFLOW, 2022).

3.3.3.1. *ImageDataGenerator*

Da biblioteca TensorFlow, utilizou-se o ImageDataGenerator que é um gerador de dados de imagens *tensor*, que gera lotes de dados com o aumento de dados em tempo real. Além disso, esse gerador faz um *loop* infinito nos dados, de modo que gera um número infinito de lotes.

3.3.4. *Keras*

O *Keras* é uma biblioteca de rede neural de código aberto escrita em *Python*. Ele é capaz de rodar em cima de *TensorFlow*, *Microsoft Cognitive Toolkit*, *R*, *Theano* ou *PlaidML*. Projetado para permitir experimentação rápida com redes neurais profundas, ele se concentra em ser fácil de usar, modular e extensível (KERAS, 2022).

Nos tópicos seguintes estão descritos os *layers*, argumentos e *optimizer* utilizados da biblioteca *Keras*.

3.3.4.1. *Optimizer* - Adam

Um otimizador é um dos dois argumentos necessários para compilar um modelo *Keras*. O otimizador aplicado neste algoritmo é o Adam, que é um método probabilístico de gradiente descendente que se baseia na estimativa adaptativa de momentos de primeira e segunda ordem.

De acordo com Kingma e Ba (2014), o método é " computacionalmente eficiente, tem pouca necessidade de memória, invariante ao reescalonamento diagonal de gradientes, e é bem adequado para problemas que são grandes em termos de dados/parâmetros ".

3.3.4.2. Convolução 2D (*Conv 2D*)

Essa camada cria um kernel de convolução que é convoluído com a entrada de camada para produzir um tensor de saídas. Se *use_bias* para *True*, um vetor de polarização é criado e adicionado às saídas. Por fim, se *activation* não for *None*, também será aplicado às saídas.

3.3.4.3. *BatchNormalization*

Esta camada que normaliza as entradas dos dados no algoritmo. A normalização em lote aplica uma transformação que mantém a saída média próxima a 0 e o desvio padrão da saída próximo a 1.

3.3.4.4. *MaxPooling2D*

Operação de sondagem máxima para dados espaciais 2D. Nessa operação realiza-se a redução da resolução da entrada ao longo de suas dimensões espaciais (altura e largura) tomando o valor máximo em uma janela de entrada (de tamanho definido por *pool_size*) para cada canal da entrada.

3.3.4.5. *Activation*

Camada de ativação que aplica uma função de ativação a uma saída, conectando as funções as saídas das camadas.

3.3.4.6. *DropOut*

A camada *DropOut* (Abandono) define aleatoriamente as unidades de entrada para 0 com uma frequência de *rate* (taxa) a cada etapa durante o tempo de treinamento, o que ajuda a evitar o *overfitting*, que representa um desempenho negativo ao utilizar os dados de teste.

Logo, esta camada funciona como um limitador para o treinamento em *loop* infinito, que outras camadas podem ocasionar, assim, quando o resultado de teste começar a se tornar negativo para o modelo, a camada abandona o teste e encerra o *loop*.

3.3.4.7. *Flatten*

O *flatten* é um nivelador de camada que não afeta o tamanho do lote.

3.3.4.8. *Dense*

Se trata da camada densa NN. *Dense* implementa a operação: *output* = (*dot (input, kernel)* + *bias*) onde *activation* é a função de ativação por elemento passada como *activationargument*, onde *kernel* é uma matriz de pesos criada pela camada e *bias* é um vetor de polarização criado pela camada (aplicável apenas se *use_bias* for *True*). Estes são todos os atributos de *Dense*.

3.3.4.9. *Callbacks*

Um retorno de chamada é um objeto que pode executar ações em vários estágios de treinamento (por exemplo, no início ou no final de uma época, número inteiro que indica a quantidade de iterações de um treinamento, antes ou depois de um único lote etc.).

Os *callbacks* são implementados neste algoritmo para:

- Gravar *logs* do *Tensorflow* após cada lote de treinamento para monitorar as métricas;
- Salvar periodicamente o modelo em disco;
- Fazer uma parada antecipada;
- Obter uma visão dos estados internos e estatísticas de um modelo durante o treinamento.

3.3.4.9.1. ModelCheckpoint

ModelCheckpoint é o *callback* aplicado no desenvolvimento desse algoritmo em conjunto com o treinamento usando *model.fit* () para salvar um modelo ou pesos (em um arquivo de pontos de verificação) em algum intervalo, para que o modelo ou pesos possam ser carregados posteriormente para continuar o treinamento a partir do estado salvo. Este *callback* é responsável por:

- Manter apenas o modelo que alcançou o "melhor desempenho" até o momento, ou para salvar o modelo no final de cada época, independentemente do desempenho.
- Definição de 'melhor'; qual quantidade monitorar e se deve ser maximizada ou minimizada.
- A frequência em que deve salvar. Atualmente, o *callback* permite salvar no final de cada época ou após um número fixo de lotes de treinamento.
- Definir se apenas os pesos são salvos ou todo o modelo é salvo.

3.3.4.9.2. EarlyStopping

Este *callback* define que o modelo deve parar de treinar quando uma métrica monitorada parar de melhorar. Nesse algoritmo, assumimos que o objetivo de um treinamento é minimizar a perda. Com isso, a métrica a ser monitorada seria 'perda' e a moda seria '*min*'. Um loop de treinamento *model.fit* () verificará ao final de cada época se a perda não está mais diminuindo, considerando o *min_delta* e *patience*, se aplicável. Uma vez que não está mais diminuindo, *model.stop_training* é marcado como *True* e o treinamento termina.

3.3.4.10. *Data argumentation*

Argumento de dados se trata de uma técnica para aumentar a diversidade de seu conjunto de treinamento aplicando transformações aleatórias (mas realistas), como rotação de imagem. Utilizou-se dos métodos *tf.image*, como:

- *tf.image.flip_left_right;*
- *tf.image.rgb_to_grayscale;*
- *tf.image.adjust_brightness;*
- *tf.image.central_crop;*
- *tf.image.stateless_random** .

Nessa etapa, as imagens são visualizadas e modificadas pelo algoritmo utilizando a técnica de *data argumentation*, distorcendo as imagens, modificando a saturação e aplicando técnicas de rotação. Assim, com a técnica o modelo obtém uma maior diversidade de imagens.

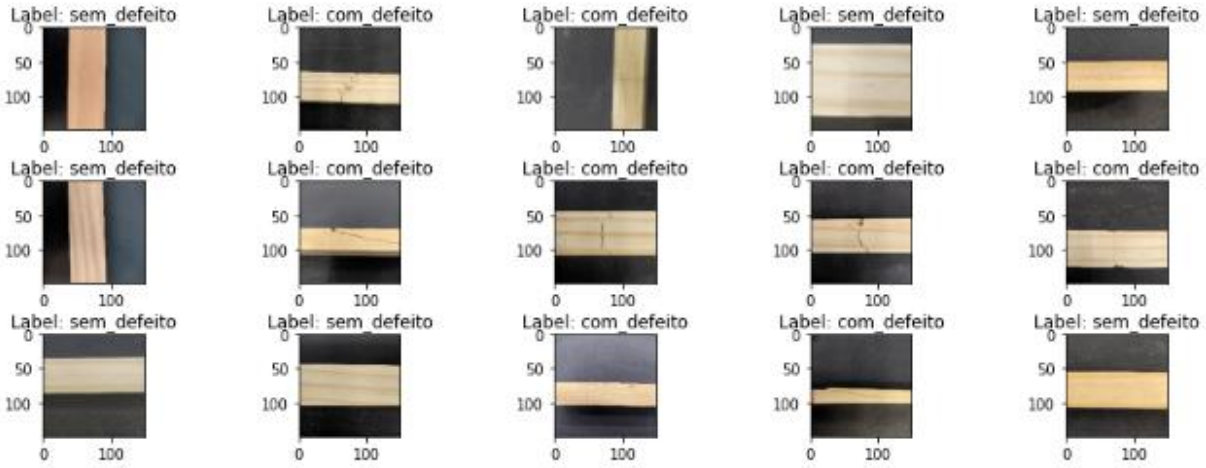

Figura 6 – Visualização das imagens do banco de dados após a aplicação do *Data argumentation*.
Fonte: A autora (2022).

3.4. Testes de qualidade de aprendizagem do modelo

Utilizou-se duas métricas de avaliação da biblioteca *Keras*, estas são *Accuracy* e *Loss*.

3.4.1. *Accuracy* (Acuracidade)

Essa métrica calcula a frequência com que as previsões são iguais aos rótulos, ou seja, a precisão das previsões. Primeiro cria-se duas variáveis locais, total e contagem, que são usadas para calcular a frequência com que *y_pred* corresponde a *y_true*. Essa frequência é finalmente retornada como precisão binária: uma matriz idempotente que simplesmente divide o total pela contagem.

Considerando isso, acurácia vai indicar a taxa de sucesso na previsão do modelo, variando seus valores entre 0 e 1, portanto, quanto mais próximo de 1 o valor da acurácia, mais preciso é o modelo.

3.4.2. *Loss* (perda)

Através da *categoricalCrossentropy*, *Loss* é calculado a perda de entropia cruzada entre os rótulos e as previsões. Portanto, o quanto o modelo perde de precisão ao longo das épocas.

Nesse sentido, a métrica variará de 0 a 1, representando uma avaliação da qualidade do treinamento do modelo em que, quanto mais próximo de 0, melhor é o treinamento do modelo.

4. RESULTADOS E DISCUSSÃO

Durante o treinamento do modelo, atingimos 8 épocas, que apresentaram os dados de perda (*loss)* da qualidade de treinamento e precisão (*accuracy*) do modelo.

Tabela 2 – Resultados das métricas de qualidade por época.

Época	**Acurácia (*accuracy*)**	***Loss***
Um (01)	0.5497	0,7139
Dois (02)	0.55413	0.7
Três (03)	0.5543	0.6878
Quatro (04)	0.5823	0.6876
Cinco (05)	0.5771	0.6874
Seis (06)	0.6874	0.5541
Sete (07)	0.5823	0.7003
Oito (08)	0.5823	0.6891

Fonte: A autora (2022).

A época é indicativo do número de iterações na matriz de dados de treinamento, a *Accuracy* é referente a precisão da análise e *loss é* indicativo da qualidade do treinamento do modelo.

Na época 1, verifica-se uma acurácia do modelo de apenas 0,5497. Ou seja, a cada imagem lida pelo modelo, a chance de que o tipo de superfície de ruptura esteja correto é de apenas de 54,97%.

Quanto à métrica *loss* que deve se aproximar de 0, obtêm-se um valor de 0,7139 que significa que o aproveitamento do treinamento do modelo não é positivo, pois este valor se aproxima mais de 1 do que de 0.

Na época 2, é notado que ambas as métricas se desenvolveram. A acurácia atingiu o nível de precisão de 55,41%, que ainda demonstra oportunidades de melhoria. E a métrica *loss* reduziu para 0,7, o que significa que apesar da melhora no treinamento, o modelo sofre uma perda de 70% na qualidade do treinamento.

Entre as épocas 3 e 4 a acurácia aumentou significamente, atingindo o valor de 58,23, no entanto *loss* reduziu apenas 0,0122 atingindo o valor de 68,76%.

Na época 6 observa-se o pico de precisão que o modelo pode atingir, que é representado pela acurácia de 58,23%. Também é nesta época que foi obtido o melhor comportamento durante o treinamento possível do modelo, representado pela *loss* de 68,27%.

Portanto, o modelo ao analisar uma imagem possui 58,23% de chance de revelar um resultado correto sobre o tipo de superfície de ruptura.

Além do mais, o modelo atingiu a *loss* de 68,27%, que não é positiva, considerando que está mais próxima de 1 do que de 0. Isso significa que o treinamento do modelo sofre uma perda de 68,27% de aprendizagem. Portanto, o modelo não está se desenvolvendo positivamente, e a cada nova época o modelo perde eficácia e oportunidade de aprendizagem, afetando, portanto, o desenvolvimento da precisão.

5. CONCLUSÃO

Os resultados de precisão do algoritmo, medidos pela métrica *accuracy*, obtêm-se o valor de 58,23%, portanto, nas condições que o modelo se encontra, apenas 58,23% das imagens lidas pelo mesmo terão seus tipos de superfície de ruptura classificados corretamente.

Quanto a métrica *loss*, que representa a qualidade do treinamento do modelo, obteve-se uma perda de 68,3% na capacidade de aprendizagem do algoritmo.

A fim de compreender o resultado obtido pelo treinamento do algoritmo, supõe-se que, a quantidade e qualidade das imagens fornecidas como banco de dados ao modelo seja um dos motivos para a limitação de aprendizagem do modelo, e portando, de sua imprecisão. Compreende-se então que existem no mínimo duas hipóteses que representam oportunidades para a aprimorar a aprendizagem do modelo.

Como primeira hipótese entende-se que um banco de dados com mais imagens representativas das variáveis, no caso os tipos de superfícies de ruptura, forneceria ao modelo uma maior quantidade de dados para o treinamento, aumentando, portanto, sua capacidade de aprendizagem, representada pela *loss*, e sua *accuracy* (acuracidade) que, influenciada pela evolução do treinamento, apresentaria resultados mais precisos.

A segunda hipótese aponta para o padrão utilizado para realização da fotografia das imagens, portanto especula-se que a utilização de um padrão de iluminação, plano de fundo, posição da peça, altura e ângulo da câmera no momento da realização da fotografia contribuiria para a melhora do resultado de treinamento e precisão do modelo. Pois, entende-se que imagens que apresentem um padrão irão prevenir que o modelo se “distraia” com diferenças causadas pela falta de técnica no momento da fotografia.

6. REFERÊNCIAS

ABREU, B. G. **Desenvolvimento de um sistema Web para utilização e gerenciamento de dados de Cupons Fiscais e Saúde**. 2016. 86 f. Monografia (Bacharelado em Sistemas de Informação) – Instituto de Ciências Exatas e Aplicada, Universidade Federal de Ouro Preto, João Monlevade-MG 2016.

ANDRIJIC, N. S. **Algoritmos, cultura e mercado de comunicação: uma análise crítica com proposta para aplicação prática**. 2019. 40 f. Monografia (Especialista em Mídia, Informação e Cultura) – Escola de Comunicações e Artes, Universidade de São Paulo, São Paulo – SP, 2019.

AMERICAN SOCIETY FOR TESTING AND MATERIALS. **ASTM D 143-94***: standard test methods for small clear speciments of timber*. West Conshohocken, PA, 2005.

ARAUJO, H. J. B. **Agrupamento das espécies madeireiras ocorrentes em pequenas áreas sob manejo florestal do Projeto de Colonização Pedro MACHADO (AC) por similaridade das propriedades físicas e mecânicas**. 2002. 168 f. Dissertação (Mestrado em Recursos Florestais. Tecnologia de Produtos Florestais) - Escola Superior de Agricultura Luiz de Queiroz, Universidade de São Paulo, Piracicaba - SP, 2002.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 7190-1: Projeto de Estruturas de Madeira**. ABNT. Rio de Janeiro, 2022.

BALLONI. C. J. V. **Caracterização física e química da madeira de *Pinus elliotti***. 2009. 41 f. Monografia (Bacharelado em Engenharia Industrial Madeireira) - Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus experimental de Itapeva, Itapeva, São Paulo, 2009.

BARCELLOS, R. **Novo Método de Mapeamento de Espaço de Cor Através de Redes Neurais Artificiais Especializada**. 2011. 121 f. Tese (Doutorado em Engenharia Elétrica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos - SP, 2011.

BREDA, A. M. M. **Classificação para madeira de *Pinus* sp. livre de defeitos a partir dos anéis de crescimento**. 2022. 72 f. Dissertação (Mestrado em Ciências Florestais) – Centro de Ciências Agrárias e Engenharias, Universidade Federal do Espírito Santo, Jerônimo Monteiro - ES, 2022.

CALONEGO, F. W.; SEVERO, E. T. D.; BRITO, A. F. ***Types of failures in thermally-modified Eucalyptus grandis wood***. Silva Lusitana, nº Especial: 153 – 161. Oeiras, Portugal, 2013.

CAMARGO, R. A.; MATOS, J. L. M **AVALIAÇÃO DA QUALIDADE DA MADEIRA DE *Pinus taeda* A PARTIR DOS ANÉIS DE CRESCIMENTO**. Universidade Federal do Paraná, Curitiba, Paraná, Brasil. 2016. Disponível em: < https://acervodigital.ufpr.br/bitstream/handle/1884/45521/RICARDO%20ARRUDA%20CAMARGO.pdf?sequence=1&isAllowed=y>. Acesso em: 20/04/2021.

CHUN, W. J. ***Core Python Programming***. Prentice Hall, 2. ed. 1120 p. United States of America, 2006.

FELISBERTO, T. Z. **Robô solucionador de Sudoku**. 2015. 144 f. Monografia (Bacharelado em Engenharia da Computação) - Universidade Federal de Santa Catarina, Araranguá, 2015.

ROVEDA FILHO, I. A. **Estudo sobre viabilidade de cultivo de *Pinus* para resinagem**. 2006. 44 f. Monografia (Bacharelado em Administração) – Universidade do Contestado, Caçador, 2006.

FLORENCIO, F. A. **Estudo comparativo do desempenho de bibliotecas para redes neurais convolucionais em diferentes microarquiteturas de GPU**. 2020. 147 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2020.

FOREST PRODUCTS LABORATORY. ***Wood handbook:*** *wood as an engineering material*. USDA, Whashigton, 1999. 463 p.

GONZALEZ, R. C.; WOODS, R. E. ***Digital Image Processing***. Pearson Education International, 3. ed. London, England, 2008.

INDÚSTRIA BRASILEIRA DE ÁRVORES - IBÁ. **Relatório anual 2020**. Brasília. Disponível em: < https://iba.org/datafiles/publicacoes/relatorios/relatorio-iba-2020.pdf>. Acesso em: 23/04/2021.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 13910**: Structural timber – Characteristic values of strength-graded timber – Sampling, fullsize testing and evaluation. Genebra, 2005. 22 p

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 16598**: Structural classification for sawn lumber, 2015.

DOBNER JÚNIOR, M.; HIGA, A. R.; ROCHA, M. P. Rendimento em serraria de toras de Pinus taeda: sortimentos de grandes dimensões. **Floresta e Ambiente**, v. 19, n. 3, p. 385-392, 2012. DOI: http://dx.doi.org/10.4322/floram.2012.053

KINGMA, D. P.; BA, J. **Adam: *a method for stochastic optimization***. Pre-print, Cornell, 2020. DOI: https://doi.org/10.48550/arXiv.1412.6980

KERAS (2022). **Keras:** The Pyrhon deep learning API. KERAS.IO, 2022. Disponível em: <https://keras.io/>. Acesso em: 12/05/2022

KOLLMANN, F. F. P.; CÔTÉ JUNIOR, W. A. ***Principles of wood science and technology I****: solid wood*. Nova York: Springer-Verlag. 1968. 592p.

LIMA, L. R. **Visão computacional aplicada para comparação analítica entre cores de duas imagens digitais**. 2021. 123 f. Monografia (Bacharel em Engenharia Elétrica) – Universidade Federal de Santa Catarina, Florianópolis, 2021.

LOPES, A; GARCIA, G. Introdução à programação. 15º ed. Rio de Janeiro: **Elsevier**, 2002.

NUMPY (2022). ***NumPy****: The fundamental package for scientific computing with Python*. NumPy, 2022. Disponível em: <https://numpy.org/>. Acesso em: 15/05/2022.

MACHADO, H. P. *Convolutional Neural Network (ConvNet/CNN) - 2*. **Medium**, 2020. Disponível em: < https://h-peixoto-m.medium.com/convolutional-neural-network-convnet-cnn-2-bbc9c01bee9b#:~:text=Uma%20rede%20neural%20convolacinal%20(Convolutional,

caracteristicas%20que%20comp%C3%B5e%20um%20objeto.>. Acesso em: 26/05/2022.

MANOVICH, L. ***The language of new media***. MIT Press, 2001.

MORESCHI, J. C. **Propriedades da madeira**. 1º ed: fev. /2.005; 4º ed: nov. /2.012. Universidade Federal do Paraná, Departamento de Engenharia e Tecnologia Florestal da UFPR, Curitiba, Paraná, 2014.

PANDAS (2022). ***Pandas***: *Python Data Analysis Library*. Pandas, 2022. Disponível em: < https://pandas.pydata.org/>. Acesso em: 19/05/2022.

RECORD, S. J. ***The Mechanical Properties of Wood****: Including a Discussion of the Factors Affecting the Mechanical Properties, and Methods of Timber Testing*. Nº 12299. USA, 2004.

SANTOS, L. L. **Estabelecimento de classes de resistência para a madeira serrada de *Pinus* sp**. 2019. 212 f. Tese (Doutorado em Engenharia Civil – Engenharia de Estruturas) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos-SP, 2019.

Sociedade Brasileira de Silvicultura (SBS). **Fatos e Números do Setor Florestal Brasileiro**. São Paulo: SBS, 2007.

TENSORFLOW (2022). **T*ensorFlow***: Uma plataforma completa de código aberto para *machine learning. TensorFlow*, 2022. Disponível em: <https://www.tensorflow.org/?hl=pt-br>. Acesso em: 15/05/2022.

ZHANGYANG, Q.; LI, B. ***Real World Masked Face Dataset***. 2021. Disponível em:< https: //github.com/X-zhangyang/Real-World-Masked-Face-Dataset>. Acesso em: 18/05/2022.

APÊNDICES

APÊNDICE A – Sobre o código do algoritmo de *image classification* para superfícies de ruptura Provocadas Pelo Ensaio de Flexão Estática na Madeira de Pinus.

Neste apêndice são apresentados trechos do código desenvolvido para o algoritmo de *image classification* para superfícies de ruptura.

No Código 1 é apresentado as bibliotecas *Numpy* e *Pandas* para processamento de dados. As bibliotecas *TensorFlow* e *Keras*, e as layers, são importadas no Código 2. A aplicação da técnica de *data argumentation* se encontra no Código 3. No Código 4, pode-se observar a implementação das *callbacks*, tais como *ModelCheckpoint* e *EarlyStopping*.

```
[ ]  import numpy as np # linear algebra
     import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
     import random

     import os
     for dirname, _, filenames in os.walk('/content/drive/MyDrive/fotos_madeira'):
         for filename in filenames:
             print(os.path.join(dirname, filename))
             break
         break
```

Código 1 – Importação das bibliotecas *Numpy* e *Pandas.*
Fonte: A autora (2022).

```
[ ]  # Importing all necessary libraries
     import tensorflow as tf
     import keras
     from keras.layers import Conv2D, BatchNormalization
     from keras.callbacks import ModelCheckpoint, EarlyStopping
     from keras import regularizers
     import sys
     from keras.preprocessing.image import ImageDataGenerator
     from keras.models import Sequential
     from keras.layers import Conv2D, MaxPooling2D
     from keras.layers import Activation, Dropout, Flatten, Dense
     from keras import backend as K
     from tensorflow.keras.preprocessing.image import load_img, img_to_array
     img_width, img_height = 224, 224
```

Código 2 – Importação das bibliotecas *TensorFlow* e *Keras*.

Fonte: A autora (2022).

```
[ ]  #Applying Data Augmentation Techniques
     datagen = ImageDataGenerator(horizontal_flip=True,
                                  vertical_flip=True,
                                  rotation_range=20,
                                  zoom_range=0.2,
                                  width_shift_range = 0.2,
                                  height_shift_range = 0.2,
                                  shear_range=0.1,
                                  fill_mode="nearest")

     testgen = ImageDataGenerator()

     datagen.fit(X_train)
     testgen.fit(X_test)
```

Código 3 – Aplicação do *data argumentation*.

Fonte: A autora (2022).

```
[ ]  #Defining the callbacks for the model such as EarlyStopping and Best Model Checkpoint
     filepath= "model_cnn_madeiras.h5"
     checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max', save_weights_only=False)

     early_stopping = EarlyStopping(monitor='val_loss',min_delta = 0, patience = 5, verbose = 1, restore_best_weights=True)

     callbacks_list = [
             checkpoint,
             early_stopping,
     #         learning_rate_reduction
         ]
```

Código 4 – Importação das *callbacks* do modelo.

Fonte: A autora (2022).