Medição da extensão em laminados de CFRP pré-esforçados no reforço de elementos de betão a partir de imagens: *machine learning versus deep learning*

Measuring strain in pre-stressed CFRP laminates for reinforcement of concrete members from images: machine learning versus deep learning

André G. Araújo
Habibu Mukhandi
Micael S. Couceiro
Jónatas Valença
Eduardo Júlio

Resumo

O reforço de estruturas de betão armado com laminados de CFRP (*Carbon Fiber Reinforced Polymer*) tem sido uma solução amplamente adotada para estruturas de grande vão. A medição do nível de deformação durante a aplicação do pré-esforço é de extrema importância, o que geralmente envolve instrumentação de elevado custo e difícil instalação. Nos últimos anos, novos métodos têm vindo a surgir, de forma a mitigar este constrangimento. Seguindo essa premissa, este trabalho apresenta uma arquitetura para a medição do nível de deformação em laminados de CFRP com base em imagens. É efetuada uma comparação com vários métodos existentes, a partir de imagens sintéticas digitalmente deformadas, considerando uma câmara de infravermelho de baixa resolução. Foram considerados métodos tradicionais de *machine learning* e de *deep learning*, tendo os últimos revelado maior precisão e exatidão.

Palavras-chave: Machine learning / Deep learning / Visão por computador / / Laminados CFRP / Monitorização de extensão

Abstract

The strengthening of reinforced concrete structures with Carbon Fiber Reinforced Polymer (CFRP) laminates has been a widely adopted solution for large span structures. The evaluation of the level of deformation during the application of pre-stress is extremely important, which generally involves instrumentation of high cost and difficult installation. In recent years, new methods have emerged in order to mitigate this constraint. Following this premise, this work presents a contact-free architecture for measuring the strain level in CFRP laminates based on images. A benchmark with several computer vision methods is performed, based on digitally deformed images considering a low-resolution infrared camera. Traditional methods of machine learning and deep learning were considered, with the latter showing greater precision and accuracy.

Keywords: Machine learning / Deep learning / Computer vision / CFRP laminates / / Strengthening RC / Strain monitoring

André G. Araújo

Investigador / CTO
IST, Universidade de Lisboa / Ingeniarius, Lda.
Lisboa, Portugal
andre.araujo@tecnico.ulisboa.pt

Habibu Mukhandi

Investigador
ISR, Universidade de Coimbra
Coimbra, Portugal
habib_mukhandi@yahoo.com

Micael S. Couceiro

CEO - Investigador
Ingeniarius, Lda - ISR, Universidade de Coimbra
Coimbra, Portugal
micael@ingeniarius.pt

Jónatas Valença

Investigador
CERIS, IST-ID, Universidade de Lisboa
Lisboa, Portugal
jonatas.valenca@tecnico.ulisboa.pt

Eduardo Júlio

Professor
CERIS, IST, Universidade de Lisboa
Lisboa, Portugal
eduardo.julio@tecnico.ulisboa.pt

Aviso legal

As opiniões manifestadas na Revista Portuguesa de Engenharia de Estruturas são da exclusiva responsabilidade dos seus autores.

Legal notice

The views expressed in the Portuguese Journal of Structural Engineering are the sole responsibility of the authors.

ARAÚJO, A. [*et al.*] – Medição da extensão em laminados de CFRP pré-esforçados no reforço de elementos de betão a partir de imagens: *machine learning versus deep learning*. **Revista Portuguesa de Engenharia de Estruturas**. Ed. LNEC. Série III. n.º 16. ISSN 2183-8488. (julho 2021) 25-38.

1 Introdução

O reforço de estruturas de betão armado com laminados pré-esforçados de CFRP *(Carbon Fiber Reinforced Polymer)* é uma solução amplamente utilizada. O pré-esforço é aplicado através de macacos hidráulicos com controle de força realizado por células de carga (Figura 1). Nos casos mais relevantes, é usada instrumentação tradicional, como transdutores e extensómetros, que permitem a monitorização direta da deformação aplicada. No entanto, é uma solução demorada e trabalhosa e, na maioria dos casos, o pré-esforço é estimado indiretamente pela força aplicada.

Muitos investigadores propõem o uso de sensores de fibra óptica para monitorizar a deformação [1]. Esta é uma solução eficaz, mas a sua precisão diminui com o tempo devido ao desgaste inerente da fibra. Além disso, é uma solução bastante dispendiosa. Outros investigadores [2] sugerem a medição de resistência elétrica dos laminados de CFRP, calculando a diferença de potencial para medir a deformação do material. Vários métodos têm também sido propostos para monitorizar a durabilidade dos laminados de CFRP através da variação da resistência elétrica ao longo do tempo. Estes métodos são económicos e utilizam as próprias fibras de carbono como sensores, sendo sistemas de deteção automática. A alteração da resistência elétrica permite medir a deformação aplicada e o desgaste das fibras de carbono. Estes métodos automáticos apresentam uma elevada precisão para monitorizar danos, quando comparados a outros métodos de medição de deformação, pois efetuam uma medição direta dos danos internos, detetando a mudança na condutividade da fibra de carbono. No entanto, como noutros métodos existentes, exigem requisitos de montagem e de configuração complexos e demorados. Além disso, implicam a instalação de elétrodos, que necessitam de estar em contacto com os laminados para que a condutividade elétrica possa ser medida entre dois pontos. Neste caso, é necessário polir o laminado para garantir que a resistência elétrica varie linearmente com a deformação imposta. Neste contexto, métodos sem contacto como os baseados na visão por computador e inteligência artificial representam uma solução atrativa e um contributo importante, pois podem apresentar várias vantagens na monitorização da aplicação do pré-esforço em laminados de CFRP, do ponto de vista técnico e económico.

A visão por computador tem sido utilizada numa grande variedade de aplicações em engenharia civil, incluindo engenharia de estruturas, onde o processamento de imagem e a fotogrametria são aplicados para monitorizar deslocamentos, deformações ou a fendilhação [3, 4]. No entanto, a instrumentação tradicional ainda é bastante utilizada, apesar de ser fortemente dependente de técnicos especializados. Atualmente, o potencial oferecido pela *machine learning* e inteligência artificial, combinado com a visão computacional, disponibiliza uma ampla gama de aplicações, e permite melhorias significativas em relação às abordagens existentes [5, 6]. Várias áreas da engenharia civil obtiveram resultados promissores utilizando *machine learning* [7,8].

Neste artigo é proposta uma arquitetura sem contacto para medir a extensão em laminados de CFRP, com base em imagens adquiridas por uma câmara de baixa resolução e custo reduzido. O objetivo principal é fornecer uma solução automática que seja

Figura 1 Aplicação de pré-esforço em laminados de CFRP com macacos hidráulicos. (Imagem cedida por S&P, Clever Reinforcement Ibérica – Materiais de Construção, Lda.)

economicamente viável, precisa, fácil de utilizar, e com elevada fiabilidade na medição da extensão nos laminados durante a aplicação de pré-esforço. A arquitetura proposta está baseada em algoritmos de visão por computador, e nesta fase do estudo o foco é a comparação de várias abordagens, aplicadas e validados em imagens sintéticas. Estas imagens foram geradas com uma resolução de 0,25 mm/pixel e considerando a adição de ruído, para simular situações reais de aquisição. De forma a tornar a simulação mais realista, foi aplicado um coeficiente de *Poisson* [9] à deformação, considerando o material homogéneo e isotrópico, o qual contrai transversalmente de forma proporcional ao alongamento na direção longitudinal. A arquitetura é avaliada considerando a exploração e a comparação de vários algoritmos, que incluem abordagens tradicionais de *machine learning* e métodos de *deep learning*.

2 Dados a analisar

2.1 Características da imagem

Foram produzidas imagens sintéticas para simular a aquisição com a câmara RealSense D435, que tem as seguintes características: distância focal de 1,93 mm; sensor com 5,252 mm × 4,478 mm (sSmm) e 1750.6667 px × 1492.67 px (sSpx), e pixel de 3 μm × 3 μm (Figura 2). A câmara está localizada em posição fronto-paralela a 150 mm do laminado, capturando imagens de 330 mm na maior dimensão do sensor, o que resulta numa resolução espacial de 4 px/mm. Estes requisitos garantem que o sensor da câmara capta objetos com 330 mm de comprimento e 50 mm de largura. As imagens sintéticas foram geradas com base nesses requisitos, com resolução de 1320 px × 200 px e um padrão de três faixas paralelas impressas. As três faixas têm 10 mm de largura e 30 mm de comprimento, e uma distância de 50 mm entre si (Figura 2).

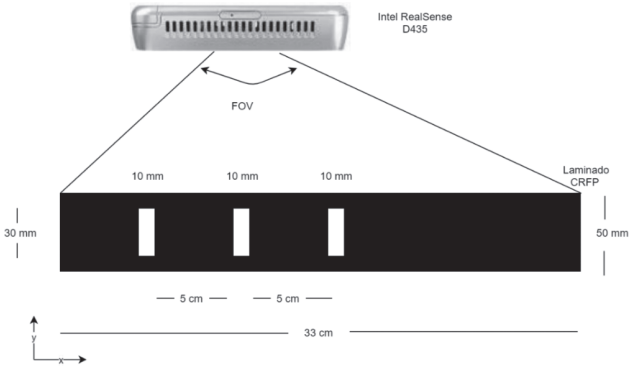


Figura 2 Configuração da aquisição de imagens do laminado com Intel RealSense D435

2.2 Deformação das imagens e ruído

As imagens foram deformadas recorrendo ao método de interpolação bi-cúbica, na direção do eixo do laminado (eixo x) para simular 101 níveis de deformação, no intervalo de 0‰ e 10‰ com incrementos de 0,1‰, num segmento de laminado com 330 mm de comprimento. Por outras palavras, o redimensionamento da imagem é usado para simular a força aplicada no laminado, multiplicando a posição de cada pixel por um fator *fx* [10]. O valor de *fx* é 1 quando a deformação imposta é 0‰, 1,001 para 1‰, e assim por diante. Adicionalmente, foi adicionado ruído às imagens sintéticas para simular as condições reais, como a existência de diferentes condições de iluminação (uma imagem pode ser parcialmente ou totalmente exposta à luz ou sombra), ruído devido à variação térmica do sensor ou à natureza do comprimento de onda infravermelha, entre outras. Os tipos de ruído gerados e aplicados

foram os seguintes: *Gaussian Noise* [11], *Salt Noise* [12], *Pepper Noise* [12], *Speckle Noise* [13], *Salt* & *Pepper Noise* [14] e *Poisson Noise* [14] (Figura 3). O ruído *gaussiano* foi adicionado para simular o efeito do ruído térmico em dispositivos eletrónicos, devido ao aquecimento do sensor da câmara. O *Salt Noise* é frequentemente adotado para simular o efeito que uma imagem exposta à luz com pixéis brilhantes em locais aleatórios. Da mesma forma, o ruído *Pepper* simula o efeito de uma imagem sem iluminação adequada, com pixéis escuros em locais aleatórios. A junção entre o ruído *Salt* & *Pepper* combina os dois efeitos, ou seja, produz uma imagem parcialmente sobre-exposta com algumas partes sub-expostas, originando a presença de pixéis escuros e claros, simultaneamente, e em locais aleatórios. O ruído de *Poisson* foi adicionado para modelar a natureza estatística das ondas eletromagnéticas, como as ondas infravermelhas, sendo o espectro fornecido pela câmara RealSense D435. As fontes infravermelhas emitem vários fotões por unidade de tempo, sendo suscetíveis à flutuação aleatória dos fotões. Isso leva a uma imagem com aleatoriedade espacial e temporal. O ruído de *Poisson*, também chamado de ruído quântico (fotão), fornece esse efeito. O ruído *Speckle* simula o efeito de um fenómeno de interferência comum, designado *speckle*, que ocorre devido à rugosidade da grande maioria das superfícies numa determinada escala do comprimento de onda do espectro eletromagnético.

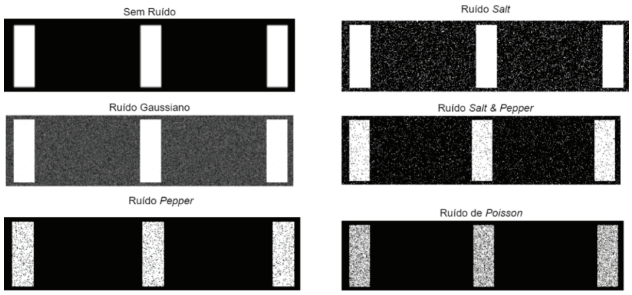


Figura 3 Imagens geradas sinteticamente, com diferentes tipos de ruídos

2.3 Treino, validação e teste

O conjunto de dados foi gerado, criando para cada nível de extensão, uma imagem sintética sem ruído (isto é, imagem original) e 29 imagens com ruído, resultando assim em 30 imagens para cada nível de deformação imposta (0‰ a 10‰ em intervalos de 0.1‰). Isto resulta num total de 3030 imagens, 30 imagens para cada um dos 101 níveis de deformação impostos. Os métodos de *machine learning* e *deep learning* adotados são baseados em três subconjuntos do conjunto principal de dados: treino, validação e teste. Assim, a partir do conjunto de dados originais (3030 imagens), a imagem original e 19 imagens com ruído, para cada nível de deformação, foram utilizadas para o treino; enquanto as restantes 10 imagens com ruído foram utilizadas para teste. Isto resulta num total de 2020 imagens para treino, e 1010 imagens para teste. Por outro lado, 10% do conjunto de treino foi utilizado para validação (202 imagens), ficando os restantes 90% para treino (1818 imagens). É de salientar que o conjunto de validação só é utilizado após cada etapa de treino dos modelos, com a intenção de avaliar qual destes tem o melhor desempenho. Os modelos voltam à etapa de treino para melhorar o seu resultado de acordo com o desempenho avaliado com o conjunto de validação. Os modelos com melhor desempenho são utilizados para a previsão de resultados dos dados não utilizados, ou seja, do conjunto de dados de teste.

3 Metodologia

3.1 Arquitetura

A Figura 4 apresenta a arquitetura geral proposta para medir o nível de deformação em laminados CFRP, usando *machine learning* e *deep learning*.

A arquitetura começa com as imagens de treino, cujas deformações impostas podem ser estimadas através de métodos tradicionais de *machine learning* ou *deep learning*. No caso da abordagem tradicional de *machine learning*, foram selecionados cinco dos métodos mais comuns presente na literatura, nos vários domínios de aplicação. Na aplicação destes métodos as imagens são sujeitas a várias etapas de pré-processamento, tais como: i) correspondência de modelos para extrair um determinado padrão entre vários; ii) eliminação do ruído das imagens; iii) extração de *features* (características específicas); e iv) estimativa da deformação dada através de *machine learning*. Por outro lado, a abordagem de *deep learning*: i) utiliza as imagens de treino e deformação imposta como dados de entrada; e ii) realiza uma estimativa em toda a extensão da deformação considerada.

3.2 *Machine learning* tradicional

A partir das imagens da Figura 2, podem ser extraídos os limites e formas das faixas padrão para obter um conjunto de cinco *features*: a largura das três faixas, o espaçamento entre a primeira e a segunda faixa, e o espaçamento entre a primeira e a terceira faixa. Foram aplicadas e avaliadas várias abordagens de *machine learning*. Para os algoritmos de machine learning tradicionais, a entrada [*x*] compreende o conjunto de *features* escolhidas manualmente pelo utilizador, ou seja, a largura de cada faixa padrão, posição e distâncias de uma faixa para outra, e os níveis de deformação impostos correspondentes são usados como variável alvo [*y*]. Ao adotar *machine learning* com métodos de regressão (regressão porque os níveis de deformação são contínuos e não discretizados), esses recursos extraídos são processados por um algoritmo de otimização, para que seja definida uma função (provavelmente uma função não linear) capaz de fornecer a relação mais adequada entre as *features* e o nível de deformação. Isso é obtido calculando os pesos que levam ao desvio mínimo entre a deformação prevista e a deformação imposta. Para os métodos tradicionais de *machine learning* foram usados os seguintes algoritmos de optimização: Regressão polinomial, *Fully Connected Neural Network* (FCNN), Regressão *Support Vector* (SVR), *Decision Tree* e *Random Forest*. Dada a conhecida suscetibilidade desses métodos ao ruído, as imagens são primeiro filtradas e, posteriormente, as features são extraídas das imagens aplicando o detetor *Canny Edge*.

Aplicação de filtros

As faixas padrão não são detetadas com a presença de ruído nas imagens. De forma a facilitar o funcionamento dos algoritmos

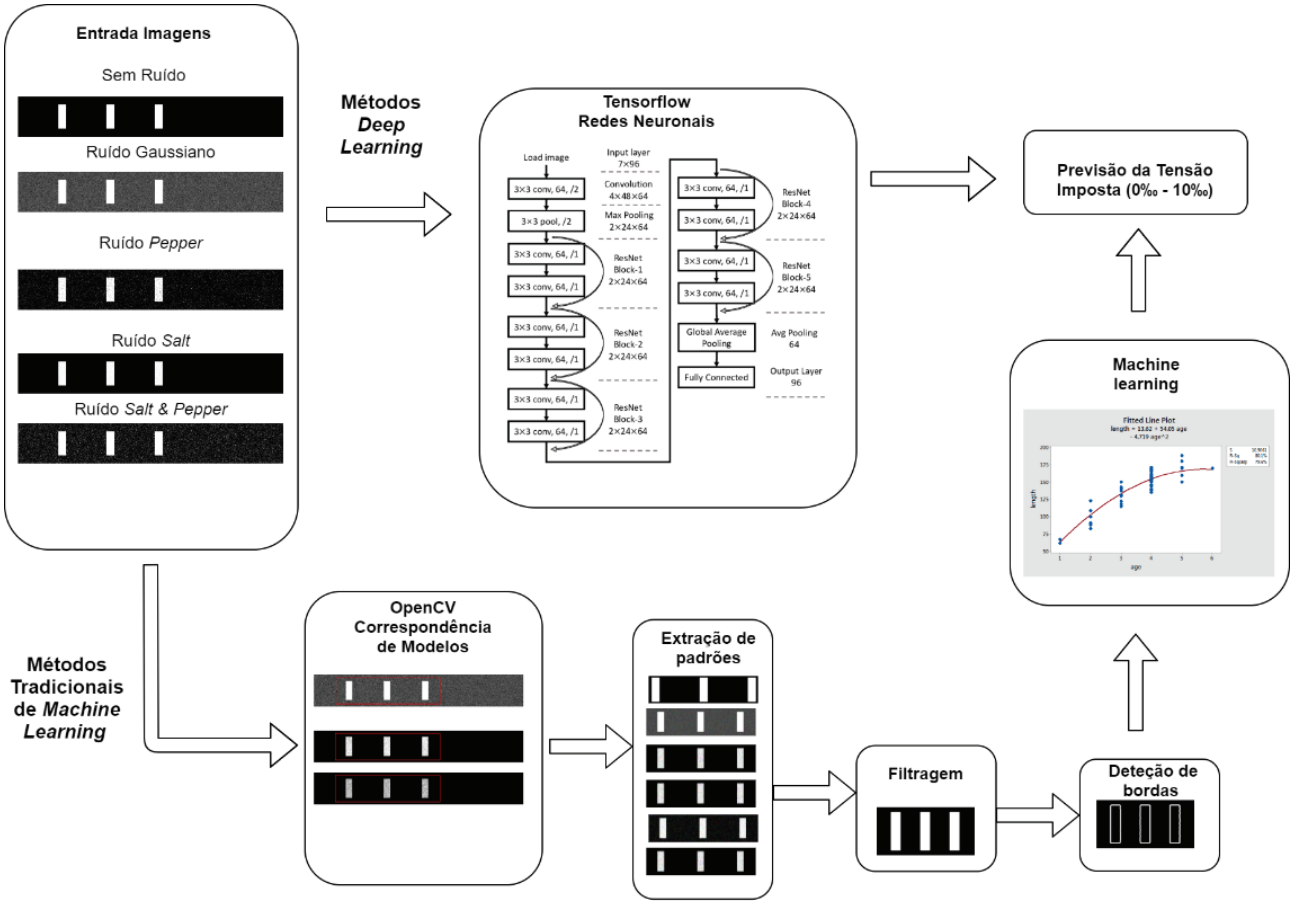


Figura 4 Esquema de arquitetura geral do sistema

de *machine learning* foram usados diferentes tipos de filtros para remover o ruído nas imagens, nomeadamente: i) *Filtro de média não local* – é baseado num princípio simples, substituir a cor de um pixel pela média ponderada das cores dos pixéis semelhantes [15]. Os pesos dados aos pixéis são calculados por uma função de similaridade, como a função gaussiana. Este método é eficaz na remoção da maioria dos tipos de ruídos nas imagens, preservando os limites dos objetos; ii) *Filtro de mediana* – calcula a mediana de todos os pixéis numa janela e o pixel central é substituído por este valor mediano. É altamente eficaz na remoção de ruído *Salt* & *Pepper* [16].

Deteção dos limites

Conforme abordado anteriormente, as *features* são extraídas das imagens usando o detetor *Canny Edge*. Na prática, um padrão é extraído usando o *Template Matching* [17] e o *Canny Edge* [18] é usado para saber a posição e a forma de cada faixa padrão. Depois de impor extensão ao laminado, o espaçamento entre as faixas e as suas larguras aumentam. A partir dessas alterações, podem ser extraídas as cinco medidas anteriormente mencionadas (Figura 2), que são usadas como entrada para um algoritmo de *machine learning* tradicional (Figura 5).

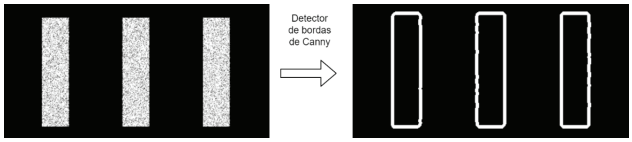


Figura 5 Aplicação do detetor de limites Canny Edge

Métodos de *machine learning* implementados

Como referido anteriormente, foram testados cinco métodos de *machine learning* para resolver o problema de regressão, considerando as cinco *features* de entrada extraídas das imagens (x_1, x_2, x_3, x_4, x_5). Espera-se que os métodos forneçam um resultado [y] que represente o nível de deformação relacionada com um determinado conjunto de *features* de entrada. De seguida são descritos resumidamente cada um dos cinco métodos adotados:

Regressão Polinomial – é um algoritmo de regressão linear com recursos polinomiais [19]. A regressão linear tenta encontrar uma relação linear entre os recursos de entrada [x] e a variável de destino ou alvo [y]. No entanto, ao introduzir características polinomiais, é possível encontrar uma função não linear capaz de mapear as

cinco *features* (x_1, x_2, x_3, x_4, x_5), com uma estimativa do nível de deformação (y_i) para um domínio, onde um relacionamento não linear pode ocorrer. Esta é a razão pela qual a regressão polinomial foi selecionada, devido à não linearidade da relação entre a variável alvo e os cinco recursos. A regressão polinomial combina as cinco *features* tais como $h(x) = \theta_0 + \theta_{1n} x_1^n + \theta_{2n} x_2^n + \theta_{3n} x_3^n + \theta_{4n} x_4^n + \theta_{5n} x_5^n + \theta_6 x_1 x_2 + \theta_7 x_1 x_2 x_3 + \theta_8 x_1 x_2 x_3 x_4 + (\ldots)$, normalmente, $n = 1,2,3,4$, onde os θ são coeficientes do modelo e $h(x)$ é o valor previsto pelo modelo de regressão polinomial.

Decision Tree – é um método supervisionado que cria um modelo que prevê o valor de uma variável alvo através da aprendizagem de heurísticas simples inferidas dos dados [20, 21]. As regras aprendidas são aplicadas na divisão de ramos de 'árvore', que dividem as *features*, minimizando a função de custo, usando divisão binária recursiva e, em seguida, usando a 'poda' de complexidade de custo mínimo para reduzir a variância do modelo. Uma árvore de decisão pode aprender uma relação não linear entre recursos e uma variável de destino sem parâmetros. Conforme abordado anteriormente, as cinco características extraídas das imagens possuem uma relação não linear com a variável alvo. O processo de localização dos hiperparâmetros ideais é denominado de ajuste de hiperparâmetros. O único hiperparâmetro a ser ajustado é a profundidade da árvore. A profundidade máxima ótima é determinada por validação cruzada, sendo a árvore ótima usada para inferência nos dados de teste.

Regressão *Random Forest* – é um meta-estimador, que se ajusta a uma série de classificadores de árvores de decisão, ou regressões, em várias sub-amostras do conjunto de dados, usando a média para melhorar a precisão preditiva e controlar a variância [22]. O *Random Forest* comporta-se como *Bootstrap Aggregation (Bagging)* quando cada árvore usa todos os recursos do conjunto de dados. Além disso, seleciona um subconjunto de recursos aleatoriamente por cada divisão de cada árvore, para evitar a correlação da árvore, num processo denominado decorrelação. Assim como acontece com as árvores de decisão, o ajuste do hiperparâmetro é usado para obter o número ideal de árvores no método e o subconjunto ótimo de recursos. Os hiperparâmetros correspondem ao número de árvores na floresta e ao número de *features* a utilizar como entrada para as árvores.

Regressão *Support Vector* – é um algoritmo de margem máxima que tenta encontrar o hiperplano $f(x)$, que melhor se ajusta a um número máximo de pontos de dados [23]. O melhor ajuste está correlacionado com o hiperplano mais próximo da maioria dos pontos que é o mais suave possível e se desvia ao máximo dos alvos [y_i] por ε. As máquinas de vetor de suporte usam um *kernel* para transformar dados de entrada, para aumentar a eficiência computacional, em vez de combinações diretas de recursos de entradas, como na regressão polinomial.

Fully Connected Neural Network (FCNN) – a arquitetura FCNN, no caso apresentado, foi criada usando o ajuste dos hiperparâmetros para definir o número óptimo de neurónios por camada, o número de camadas ocultas, a taxa de aprendizagem da rede (η), e o algoritmo de otimização. Uma arquitetura FCNN ideal, criada pela combinação dos hiperparâmetros ideais mencionados, foi utilizada para fazer as previsões nos dados de teste. Como anteriormente, os dados de entrada são as cinco *features* extraídas de cada imagem e os valores alvo são os níveis de extensão de uma imagem.

3.3 *Deep learning*

Ao contrário das abordagens tradicionais de visão por computador, o *deep learning* para regressão é alimentado diretamente pelas imagens, ao invés de uma seleção de recursos nas imagens [24, 25]. Por outras palavras, nos algoritmos supervisionados de *deep learning* adotados, as imagens são usadas como dado de entrada, [x], e os seus correspondentes níveis de deformação são usados como variável alvo [y]. O algoritmo de *deep learning* é então responsável por extrair automaticamente as *features* através dos seus núcleos de convolução, também conhecidos como pesos, e estimar uma função capaz de relacionar essas *features*, multiplicadas pelos respetivos pesos, às deformações impostas ao laminado. Todas as arquiteturas de *deep learning* apresentadas usam redes neuronais convolucionais (CNNs) bidimensionais, camadas de *dropout*, camadas de *maxpool*, camadas de normalização, e uma camada totalmente conectada como blocos de construção. Os métodos adotados foram:

GoogLeNet – é uma arquitetura de rede neuronal convolucional desenvolvida pela Google que usa módulos de iniciação, ou seja, diferentes tipos de tamanho de *kernel*, como 1 × 1, 3 × 3, 5 × 5, e uma camada *MaxPooling* no mesmo módulo por concatenação [26]. Apesar do *GoogLeNet* ser muito profundo, ele usa um número menor de parâmetros (cerca de 6 milhões de parâmetros para torná-lo computacionalmente menos dispendioso) do que muitas outras arquiteturas de redes neuronais convolucionais. Por outro lado, o *GoogleNet* é muito poderoso devido ao fato das camadas iniciais incluírem todos os tamanhos de *kernel* e a camada *MaxPool* num único módulo inicial. Neste artigo, a função de ativação da camada final foi substituída por uma função de ativação linear com menor desvio, para fazer com que o *GoogLeNet* execute tarefas de regressão.

ResNet – permite treinar redes neuronais muito profundas, com mais de cem camadas, sem ter problemas de gradiente de desaparecimento [26]. Este problema ocorre quando os pesos das camadas convergem para zero e o gradiente fica plano (mais detalhes em [28]). A arquitetura *ResNet* consegue isso por meio de uma técnica que ignora a conexão. Tal como no *GoogLeNet* com regressão, a função de ativação da camada final foi substituída por uma função de ativação linear com redução do desvio, e foram usadas 34 camadas (também conhecidas como *ResNet34* com regressão).

3.4 Resultados

3.4.1 *Machine learning* tradicional

O pré-processamento de dados com o detetor *Canny Edge*, que remove o ruído usando o *kernel Gaussiano*, foi bem sucedido na remoção dos ruídos *Poisson* e *Speckle*. No entanto, não diminui satisfatoriamente o ruído *Pepper*. Além disso, o detetor *Canny Edge* reduz o ruído *Salt e Salt* & *Pepper* de forma inadequada e não preserva os limites dos objetos corretamente. Portanto, o filtro de média não local e o filtro de mediana também foram aplicados às imagens para remover o ruído. O primeiro permite remover o ruído enquanto preserva os limites dos objetos, algo crucial para a extração das faixas padrão no caso apresentado. Para a regressão

polinomial, o ajuste realizado para encontrar o hiperparâmetro com o menor RMSD no conjunto de validação, gerou um grau polinomial ideal de 2 e um desvio quadrático médio e RMS, no conjunto de teste, de 0,3498‰ e 0,5914‰, respetivamente.

Para a regressão *Decision Tree*, o procedimento de ajuste para encontrar o hiperparâmetro com o menor desvio no conjunto de validação definiu uma profundidade ideal da árvore de 10. O desvio quadrático médio de 0,2609‰ e RMSD de 0,5108‰ foram alcançados no conjunto de teste.

Para regressão *Random Forest*, o procedimento de ajuste de hiperparâmetro para encontrar o número ideal de 'árvores' e subconjunto de *features* com o mínimo RMSD no conjunto de validação, definiu um número ideal de 100. Isso resulta num desvio quadrático médio de 0,2560‰ e RMSD de 0,5060‰ no conjunto de teste.

Para a regressão *Support Vector* a combinação de hiperparâmetro ideal, que fornece o mínimo RMSE no conjunto de validação, é atingida para $C = 50$, *coef* $0 = 5$, $\varepsilon = 0{,}3$, $\gamma = 0{,}1$, e um *kernel* linear, que resulta num desvio quadrático médio no teste de 0,5925‰ e RMSD de 0,7698‰.

Para a regressão *Fully Connected Neural Network* (FCNN), o hiperparâmetro com o mínimo RMSD no conjunto de validação é atingido para 4 e 20 neurónios por cada camada oculta. O ajuste realizado definiu ainda uma taxa de aprendizagem (η) de 0,003, com *Stochastic Gradient Descent* como otimizador. Neste caso forneceu um desvio quadrático médio e RMSD, no conjunto de teste, de 0,4050‰ e 0,6364‰, respetivamente.

3.4.2 *Deep learning*

No caso da aplicação de *deep learning*, não houve necessidade de ajustar os hiperparâmetros devido à disponibilidade de arquiteturas predefinidas por outros investigadores, nomeadamente da Google, Facebook, Microsoft, universidades e outras entidades de investigação.

Para a arquitetura *GoogLeNet*, o desvio quadrático médio no conjunto de teste foi de 0,0082‰ e RMSD é 0,0910‰. No caso da arquitetura *ResNet34*, um desvio quadrático médio de 0,0032‰ e um RMSD de 0,057‰ foram obtidos no conjunto de teste com uma elevada confiança de 99,94%.

3.4.3 Comparação de resultados

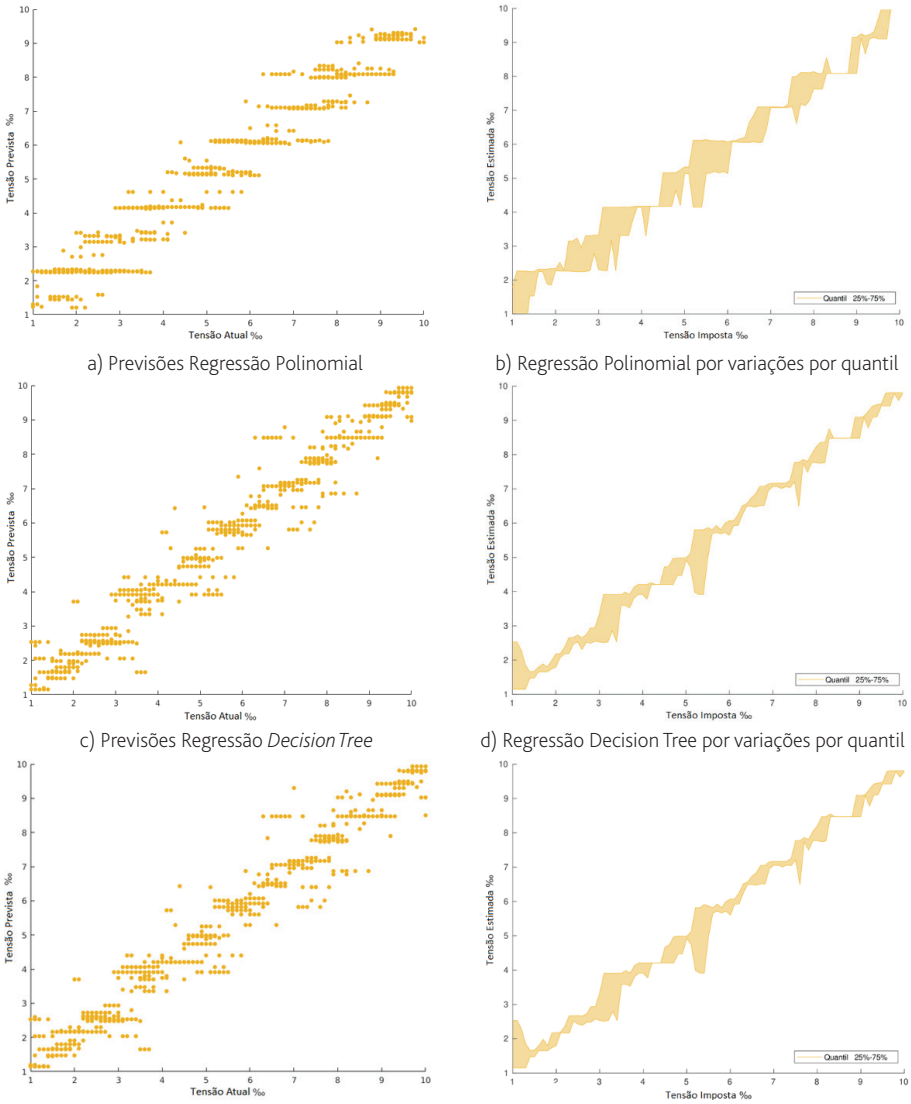
A Tabela 1 resume os resultados alcançados e detalhados da Figura 6 à Figura 9. Avaliando e comparando a deformação imposta e a estimada, calculando os desvios quadrático médio e RMS; e a precisão analisando a dispersão dos resultados, calculando a variância explicada e a covariância.

Os métodos tradicionais de *machine learning* mostram maior desvio na estimativa da deformação, o que resulta em menores variâncias explicadas entre os níveis de deformação estimados [$\hat{y}$], e os níveis de deformação impostos [y]. Isso é também claramente evidente na dispersão em quartis, entre 25% e 75%, dos resultados obtidos na Figura 9.

Por outro lado, os algoritmos de *deep learning*, como *ResNet* e *GoogLeNet*, têm melhor desempenho no conjunto de validação e, portanto, no conjunto de teste do que os métodos tradicionais de *machine learning* para visão por computador. O algoritmo *ResNet34* conseguiu superar todos os algoritmos avaliados neste artigo com RMSD de 0,057‰ no conjunto de teste. Além disso, o elevado nível de confiança de mais de 99,9% nas novas observações apresenta menor dispersão de resultados (Figura 8), quando comparados aos demais. A comparação entre a dispersão de resultados dos métodos tradicionais de *machine learning* e o *Resnet34* é apresentada nos gráficos da Figura 8. Neste, as estimativas dos métodos tradicionais de *machine learning* têm claramente maior dispersão na previsão da extensão em comparação com os métodos de *deep learning*, que são mais estáveis e confiáveis. O RMSD associado aos métodos de *deep learning* permanece reduzido em todos os níveis de deformação, conforme apresentado na Figura 9.

Tabela 1 Comparação de resultados

	Desvio quadrático médio (‰)	Desvio da raiz quadrática média (RMSD) (‰)	Variação explicada	Covariância ($x10^{-6}$)
Regressão Polinomial	0,3498	0,5914	0,9493	6,5199
Regressão *Decision Tree*	0,2609	0,5108	0,9620	6,6470
Regressão *Random Forest*	0,2560	0,5060	0,9631	6,6327
Fully Connected Neural Network	0,5925	0,7698	0,9141	6,2753
Regressão *Support Vector*	0,4050	0,6364	0,9411	6,7609
Regressão + *GoogleNet*	0,0082	0,0910	0,9989	7,0399
Regressão + *ResNet*	**0,0032**	**0,0570**	**0,9996**	**7,1225**



a) Previsões Regressão Polinomial

b) Regressão Polinomial por variações por quantil

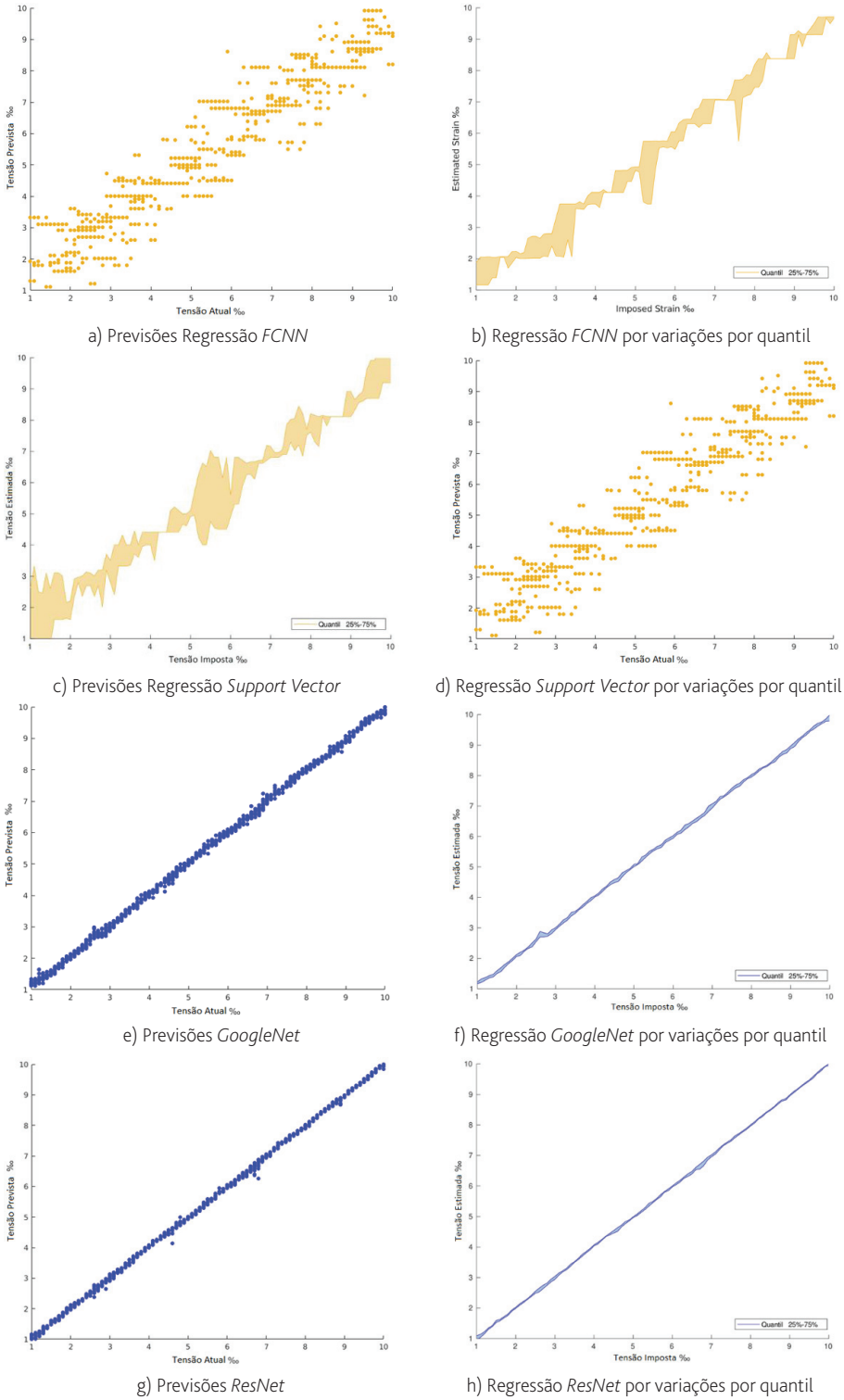
c) Previsões Regressão *Decision Tree*

d) Regressão Decision Tree por variações por quantil

e) Previsões Regressão *Random Forest*

f) Regressão *Random Forest* por variações por quantil

Figura 6 Variação de previsões fornecida pelos vários métodos



a) Previsões Regressão *FCNN*

b) Regressão *FCNN* por variações por quantil

c) Previsões Regressão *Support Vector*

d) Regressão *Support Vector* por variações por quantil

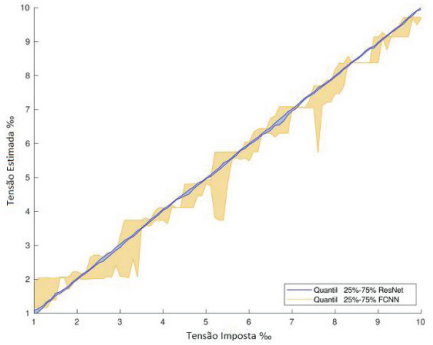
e) Previsões *GoogleNet*

f) Regressão *GoogleNet* por variações por quantil

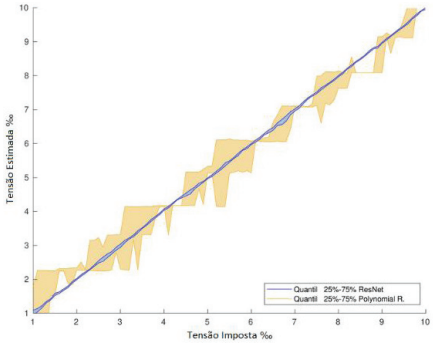
g) Previsões *ResNet*

h) Regressão *ResNet* por variações por quantil

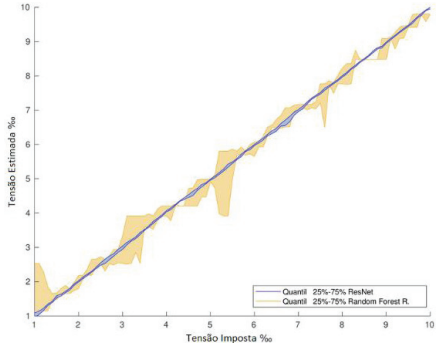
Figura 7 Variação de previsões fornecida pelos vários métodos



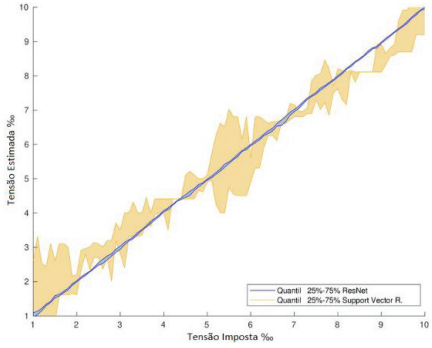
a) Regressão *FCNN* vs *ResNet*



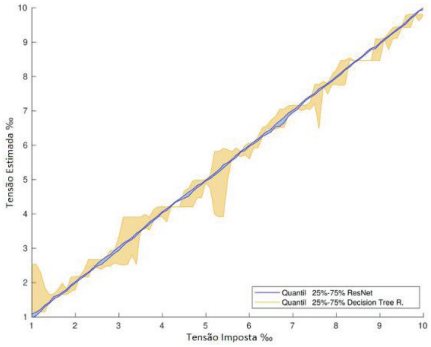
b) Regressão Polinomial vs *ResNet*



c) Regressão *Random Forest* vs *ResNet*

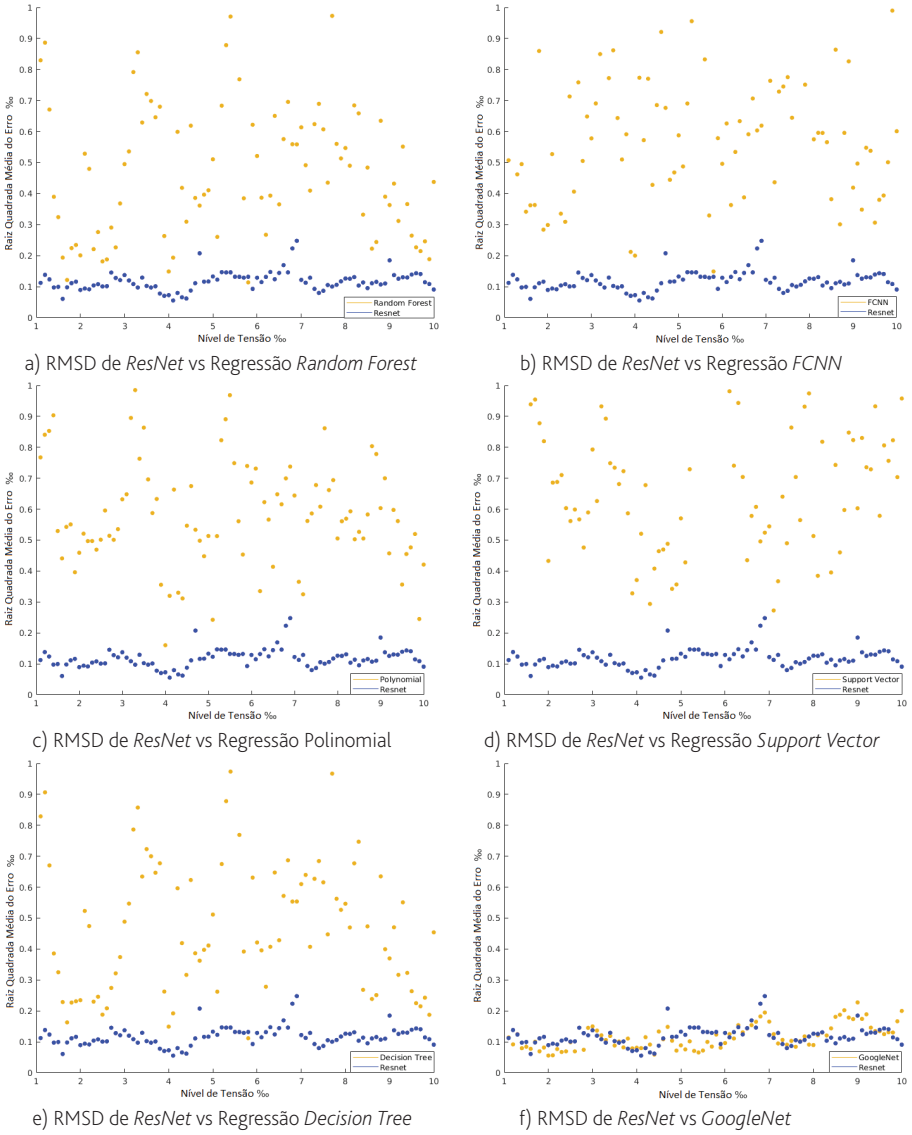


d) Regressão *Support Vector* vs *ResNet*



e) Regressão *Decision Tree* vs *ResNet*

Figura 8 Comparação *ResNet* com os restantes métodos



a) RMSD de *ResNet* vs Regressão *Random Forest*

b) RMSD de *ResNet* vs Regressão *FCNN*

c) RMSD de *ResNet* vs Regressão Polinomial

d) RMSD de *ResNet* vs Regressão *Support Vector*

e) RMSD de *ResNet* vs Regressão *Decision Tree*

f) RMSD de *ResNet* vs *GoogleNet*

Figura 9 Comparação do RMSD entre *ResNet* e restantes métodos

Por fim, importa referir que o tempo de teste dos métodos tradicionais de *machine learning* numa máquina local com especificações de CPU Intel (R) Core (TM) i5-7500T @2,70 GHz é de 0,15 segundos. Os algoritmos de *deep learning* precisam ser executados pelo menos com 200 a 800 *epoch* durante o treino, para encontrar as soluções ideais. O que no caso avaliados se traduziu em 80 segundos para treino por *epoch* numa unidade de processamento de tensor (TPU) de 8 núcleos, e cerca de 0,1 segundos de teste por imagem e 2 segundos na máquina local. O RMSD tende para zero com o aumento do número de *epoch* durante o treino.

4 Conclusões

Neste artigo é apresentada uma arquitetura sem contacto, baseada no processamento de imagens, para medir o nível de deformação em laminados de CFRP, aquando da aplicação de pré-esforço. A arquitetura implementada visa a comparação com vários métodos existentes, as quais foram avaliadas para imagens sintéticas que simulam, de forma o mais realista possível, a aquisição com uma câmara de infravermelho de baixa resolução. Foram testadas e comparadas duas abordagens, usando *machine learning* tradicional e *deep learning*.

A abordagem de *deep learning*, ainda que computacionalmente mais exigente, é aquela que fornece o método mais fiável, preciso e com elevada confiança para medir a evolução do nível de deformação nos laminados durante a aplicação de pré-esforço. Os resultados sugerem o uso da arquitetura de *deep learning ResNet34*, devido ao menor desvio quadrático médio (RMSD) entre a deformação prevista e a deformação imposta. Algoritmos de *deep learning* como *ResNet* têm a maior 'variância explicada', 0,9996, entre deformação imposta e estimada, muito próxima da variância explicada perfeita de 1; e a covariância mais elevada, $7{,}12 \times 10^{-6}$. A arquitetura é simples, mas robusta ao ruído nas imagens e tem um tempo de inferência relativamente curto, de cerca de um segundo na CPU da máquina local utilizada. Os métodos de *deep learning* requerem treino na *cloud* devido à elevada utilização de memória da GPU e aos grandes requisitos de energia computacional. No entanto, pode ser utilizada uma CPU para inferência. Também foram exploradas redes mais profundas, como *ResNet150*, ou seja, *ResNet* com 150 camadas, mas o tempo de treino por *epoch* é de 1 hora numa unidade de processamento de tensor de 8 núcleos, fornecida pela Google Cloud, em comparação com 580 segundos por *epoch* obtidos pela arquitetura aqui apresentada. É presumível que redes mais profundas tenham um desempenho ligeiramente melhor do que a arquitetura proposta, mas com maior custo computacional.

Referências

[1] Webb, G. *et al.* – "Analysis of fiber-optic strain-monitoring data from a prestressed concrete bridge". Journal of Bridge Engineering 2017;22(5):05017002.

[2] Todoroki, A.; Ueda, M.; Hirano, Y. – "Strain and damage monitoring of CFRP laminates by means of electrical resistance measurement". Journal of Solid Mechanics and Materials Engineering 2007; 1(8): 947-974.

[3] Valença, J.; Júlio, E. – "Crack-Dam: the scale-up of a method to assess cracks on concrete dams by image processing". The case study of Itaipu Dam, at the Brazil–Paraguay border. Journal of Civil Structural Health Monitoring 2018;8(5):857-866.

[4] Godinho, L.; Dias-da Costa, D.; Valença, J.; Areias, P. – "An ecient technique for strain recovery from photogrammetric data using meshless interpolation". Strain 2014;50(2):132-146.

[5] Salehi, H.; Burgueno, R. – "Emerging artificial intelligence methods in structural engineering". Engineering structures 2018;171:170-189.

[6] McLaughlin, E.; Charron, N.; Narasimhan, S. Automated – "Defect Quantification in Concrete Bridges Using Robotics and Deep Learning". Journal of Computing in Civil Engineering 2020;34(5):04020029.

[7] Hu, Y.; Castro-Lacouture D. – "Clash relevance prediction based on machine learning". Journal of Computing in Civil Engineering 2019;33(2):04018060.

[8] Nitsche, P.; Stütz, R.; Kammer, M.; Maurer, P. – "Comparison of machine learning methods for evaluating pavement roughness based on vehicle response". Journal of Computing in Civil Engineering 2014;28(4):04014015.

[9] Greaves, G.N.; Greer, A.L.; Lakes, R.S.; Rouxel, T. (2011). Poisson's ratio and modern materials. Nature materials, 10(11), 823-837.

[10] Park, H.; Park, Y.; Oh, S.K. – "L/M-fold image resizing in block-DCT domain using symmetric convolution". IEEE transactions on image processing 2003;12(9):1016-1034.

[11] Slepian, D. – "The one-sided barrier problem for Gaussian noise". Bell System Technical Journal 1962;41(2):463-501.

[12] Chan, R.H.; Ho, C.W.; Nikolova, M. – "Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization". IEEE Transactions on image processing 2005;14(10):1479-1485.

[13] Racine, R. *et al.* – "Speckle noise and the detection of faint companions". Public of the Astronomical Society of the Pacic 1999;111(759):587.

[14] Le, T.; Chartrand, R.; Asaki , T.J. – "A variational approach to reconstructing images corrupted by Poisson noise". Journal of mathematical imaging and vision 2007;27(3):257-263.

[15] Buades, A.; Coll, B.; Morel, J.M. – "A non-local algorithm for image denoising". In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2 IEEE; 2005. p. 60-65.

[16] Lu, C.T.; Chou, T.C. – "Denoising of salt-and-pepper noise corrupted image using modified directional-weighted-median filter". Pattern Recognition Letters 2012;33(10):1287-1295.

[17] Brunelli, R. – "Template matching techniques in computer vision: theory and practice". JohnWiley & Sons; 2009.

[18] Bao, P,.; Zhang, L.; Wu, X. – "Canny edge detection enhancement by scale multiplication". IEEE transactions on pattern analysis and machine intelligence 2005;27(9):1485-1490.

[19] Montgomery, D.C.; Peck, E.A.; Vining, G.G. – "Introduction to linear regression analysis", vol. 821. JohnWiley & Sons; 2012.

[20] Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. – "Classification and regression trees". Belmont, CA: Wadsworth. International Group 1984;432:151-166.

[21] Safavian, S.R.; Landgrebe, D. – "A survey of decision tree classier methodology". IEEE transactions on systems, man, and cybernetics 1991;21(3):660-674.

[22] Segal, M.R. – "Machine learning benchmarks and random forest regression" 2004;

[23] Drucker, H. *et al.* – "Support vector regression machines". In: Advances in neural information processing systems; 1997. p. 155-161.

[24] Goodfellow, I.; Bengio, Y.; Courville, A. – "Deep Learning: Adaptive computation and machine learning". Excerpt from Chapter 9. MIT Press; 2016.

[25] LeCun, Y. *et al.* – "Backpropagation applied to handwritten zip code recognition". Neural computation 1989;1(4):541-551.

[26] Zhong, Z.; Jin, L.; Xie, Z. – "High performance oine handwritten chinese character recognition using googlenet and directional feature maps". In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR) IEEE; 2015. p. 846-850.

[27] Szegedy, C.; Lofie, S.; Vanhoucke, V.; Alemi, A.A. – "Inception-v4, inception-resnet and the impact of residual connections on learning". In: 31st AAAI conference on articial intelligence; 2017.

[28] Hochreiter, S. – "The vanishing gradient problem during learning recurrent neural nets and problem solutions". International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 1998;6(02): 107-116.