UNIVERSIDADE FEDERAL DE VIÇOSA

DEPARTAMENTO DE FÍSICA

REDES NEURAIS CONVOLUCIONAIS PARA PROBLEMAS DE OTIMIZAÇÃO TOPOLÓGICA

Humberto Sousa Martins – 93724

Prof. Álvaro Neves (Orientador)

Viçosa – Minas Gerais – Brasil

Julho – 2022

Agradeço à minha mãe Angela e minha família,
pelo apoio incondicional nessa longa viagem de quase 7 anos.
À minha companheira, amor e melhor amiga Sara,
por segurar minha mão em todos os momentos.
Ao meu colega João Vítor Neves,
pelo trabalho conjunto e pelos muitos insights.
Ao meu orientador Álvaro Neves
Pela sua paciência, atenção e guiança.
À UFV e ao DPF e aos meus colegas de curso,
Pelos momentos infindáveis de alegrias, frustrações, amizades e trabalho duro.

RESUMO

A área de otimização topológica estuda como distribuir espacialmente o material que constitui uma estrutura de modo a maximizar a sua rigidez mecânica, satisfazendo certas condições de contorno. Para tal, métodos tradicionais calculam a distribuição de tensões internas da estrutura, chamada sensibilidade, e buscam reduzi-la removendo e adicionando material em passos discretos iterativamente. Devido à alta demanda computacional dos métodos atualmente empregados, o problema de reduzir o tal custo ocupa uma posição de destaque nessa área. O presente trabalho investiga a possibilidade de fazê-lo utilizando um método de inteligência artificial: o aprendizado de máquina com redes neurais. A ideia fundamental é visualizar o problema de otimização topológica como um de segmentação semântica, resolvido treinando redes neurais para previsão da topologia otimizada por um método iterativo tradicional, a partir de dados pré-otimizados de estruturas geradas pelo método iterativo, com menor custo computacional. Foram usadas redes convolucionais baseadas na U-Net, celebrizada na segmentação semântica de imagens médicas. Testamos inovações conceituais em relação à trabalhos semelhantes da área: o uso da sensibilidade condicionada e de seu gradiente espacial como entrada para as redes, contrário ao uso da topologia e seu gradiente temporal, amplamente empregados na literatura, assim como emprego da entropia binária cruzada como função custo, sem os termos aditivos. Os dados para treinamento e teste da rede foram gerados resolvendo 10 mil problemas aleatórios de otimização de estruturas mecânicas pelo método BESO. Foi de grande interesse testar a performance da rede para exemplos pouco ou nada pré-otimizados pelo método tradicional, situação de baixa performance para redes que usam topologia como entrada. Os resultados mostram que o uso de sensibilidades como entrada gerou resultados de alta fidelidade com as estruturas originais, para 5 iterações da pré-otimização obtendo acurácia binária de 92,1% em dados de validação. Mais impressionante foi a performance em dados com 0 iterações de pré-otimização, em que a rede obteve os mesmos 92,1% de acurácia binária, resultado muito superior ao caso de redes que usam topologia como entrada. A rede pode performar suas previsões em um tempo 6 ordens de grandeza menores que os métodos iterativos, resultando em um significativo ganho computacional à custa de baixa perda de precisão.

ABSTRACT

Topology optimization is an area of study whose objective is to spatially distribute material that compose a structure in a way to maximize its mechanical resilience, given boundary conditions. For such, traditional methods calculate the structure's internal tensions distribution, here named sensibility, and seek to minimize it by iteratively removing and adding material in discrete steps. Due to high computational demand of such method, the problem of reducing its computational cost occupies a prominent position in this area. This present work investigates a possibility of performing it using an artificial intelligence method: machine learning with neural networks. The fundamental idea is visualizing the problem of topology optimization as one of semantic segmentation, and solve it by training networks to predict the optimized topology of a traditional iterative method, from data of a structure preliminarily optimized by said method, with lowered computational cost. For that, convolutional neural networks will be used, based on the U-Net model, now famous for its breakthrough the field of semantic segmentation of medical images. We will test important conceptual innovations regarding related works in the field: the use of conditioned sensitivity and its special gradient inputs, in contrast to more conventional topology and its temporal gradient, and binary cross-entropy as a cost function without additive terms. The dataset for training and validation was generated by solving 10.000 randomly generated problems with the BESO method. It was of great interest to test the network's performance on data with few pre-optimization steps, or even none at all, in which networks that use topology as input show poor performance. Results show that using sensibility as input achieved results of high fidelity with the original structures, for 5 pre-optimization steps obtaining a binary accuracy of 92.1% in validation data. Most impressive was its performance in data from 0 pre-optimization steps, in which the network maintained a binary accuracy of 92.1%, a result greatly superior to those networks using topology as input. The network was able to complete the optimization in times 6 orders of magnitude shorter then those of traditional methods, resulting in significant computational gain in exchange for small loss of precision.

SUMÁRIO

1. OBJETIVO

O propósito deste trabalho é estudar a viabilidade de usar métodos de aprendizado de máquina com redes neurais para reduzir o custo computacional da otimização topológica de estruturas mecânicas. Serão usadas redes convolucionais profundas, baseadas na U-Net, em como em alguns trabalhos na literatura da área. Contudo, serão estudadas inovações conceituais significativas, como o uso da sensibilidade condicionada como entrada para as redes e o da entropia binária cruzada como função custo, sem os termos aditivos empregados na literatura. O ponto central desta investigação será comparar o desempenho da nossa abordagem com o do célebre trabalho de Sosnovik e Oseledets (2019).

2. INTRODUÇÃO

A otimização topológica é uma área vibrante da engenharia, com aplicabilidade ampla e crescente. Ela endereça o problema de como distribuir o material que constitui uma estrutura, de modo a otimizar o seu desempenho, respeitando algumas limitações. Um problema clássico é maximizar a rigidez mecânica de uma estrutura, como uma ponte, dadas as forças externas que atuam sobre ela, a sua extensão e outras condições de contorno, como a quantidade de material que se permite utilizar. Nesse caso, o processo de otimização determina a distribuição do material no espaço de projeto, visando minimizar a energia elástica de deformação (sensibilidade total) armazenada na estrutura. Os métodos de otimização estrutural são aplicados a projetos com diferentes objetivos e limitações como, por exemplo, minimizar a temperatura um circuito integrado, ou maximizar a rigidez de uma torre e concomitantemente a sua menor frequência de ressonância e ainda a separação entre as várias frequências de ressonância.

Os métodos evolucionários destacam-se entre os correntemente utilizados na otimização topológica. Neles, partindo-se de uma distribuição inicial de material e de condições de contorno, usa-se o método de elementos finitos para calcular grandezas de interesse para o problema, como as distribuições de tensão, deformação, temperatura ou a energia elástica de deformação da estrutura. Tendo em vista o objetivo do processo de otimização, os métodos evolucionários indicam uma pequena modificação na topologia da estrutura em questão. O processo segue ciclicamente, calculando-se as propriedades de interesse e efetuando-se novas alterações na distribuição de material.

O custo computacional dos métodos evolucionários em problemas realísticos é elevado, frequentemente proibitivo. De fato, a redução dele é um tema de pesquisa de importância capital. Recentemente, alguns trabalhos, incluindo este, têm estudado a possibilidade de utilizar o aprendizado de máquina com redes neurais para reduzir drasticamente o processamento envolvido na otimização topológica. A ideia envolve o treinamento de uma rede neural, por meio do aprendizado supervisionado. Nele, a rede neural é treinada, isso é, têm seus parâmetros internos ajustados com base em exemplos de estruturas otimizadas. A esperança é que após o treinamento a rede, ao receber na sua entrada a descrição de uma estrutura parcialmente otimizada (por poucas iterações de um método tradicional), possa prever corretamente a distribuição de massa da estrutura completamente otimizada. É importante destacar que a otimização pela rede treinada se dá em uma única etapa, isso é, sem iterações.

Este trabalho baseia-se na contribuição seminal de Sosnovik [1]. Continuando o trabalho de Neves [2], usamos redes neurais convolucionais treinadas em exemplos advindos de métodos evolucionários e, como maior inovação, têm como entrada a sensibilidade.

3. REVISÃO BIBLIOGRÁFICA

3.1. Motivação

Otimização topológica é o estudo de criar estruturas otimizadas para fins e restrições específicas. A figura 3.1 é uma boa ilustração de tal, em que o chassi da motocicleta foi otimizado topologicamente para ter menor massa preservando a maior resistência mecânica possível, uma subclasse de problemas que chamaremos de "otimização estrutural". As restrições específicas nesse problema são os pontos de encaixe com demais peças e os espaços internos necessários, também chamadas de condições de contorno.

Figura 3.1: motocicleta com chassi otimizado topologicamente para menor massa e maior rigidez mecânica, restrito às suas condições de contorno. Retirado de [3]

A evolução dos seres vivos, através de seleção natural, é o fundamental exemplo de otimização topológica. A evolução, através de pequenos passos, otimiza formas, comportamentos e processos em todos os seres vivos. É evidente, na figura 3.2, uma certa semelhança qualitativa entre as formas de um esqueleto de pepino do mar, otimizado para maior eficiência de seus escassos recursos, e a moto apresentada na figura 3.1. De maneira semelhante, os métodos computacionais disponíveis tradicionalmente para realizar este trabalho imitam a evolução, melhorando uma estrutura em pequenos passos, portanto são chamados de métodos evolutivos.

Figura 3.2: esqueleto de pepino do mar, apresentando semelhanças qualitativas à motocicleta da figura anterior. Retirado de [4]

Tal estudo se torna cada dia mais relevante devido à recente capacidade técnica e industrial de realização de tais estruturas otimizadas no mundo real, acompanhando saltos tecnológicos nas áreas de impressão 3D e manufatura aditiva em grande escala e eficiência. Outras áreas tem igual relevância e aplicação à otimização estrutural: Otimização de fluxo de fluídos em componentes aeroespaciais, peças otimizadas para dissipação de calor da escala de semicondutores à automóveis. Dentre áreas que se beneficiam imediatamente, e por vezes são pioneiras no uso de estruturas otimizadas, se destacam a engenharia mecânica, civil, automotiva e aeroespacial, robótica e arquitetura, assim como design e até moda.

3.2. Método de elementos finitos

O método de elementos finitos [5] busca resolver sistemas de equações parciais com valores de contorno, em que o espaço contínuo do domínio é dividido em partes discretas, chamadas elementos finitos. Tal método possui aplicações em problemas de condução de calor, dinâmica de fluidos, e no caso abordado neste trabalho, otimização topológica. No último, uma estrutura planar elástica é subdividida em uma rede de elementos triangulares, nos quais os vértices são pontos adimensionais rígidos chamados “nós”, que interagem apenas através das arestas que se comportam efetivamente como molas. As forças atuantes em um determinado nó dependem então de forças externas aplicadas a ele, sua deformação inicial e de seu deslocamento em relação aos demais, que cria uma tensão nas ligações de acordo com as características elásticas do material. Estas relações podem ser expressas através da expressão (1):

$$\boldsymbol{K} * \boldsymbol{u} = \boldsymbol{f} \tag{1}$$

Dada uma matriz de rigidez $\boldsymbol{K}$, calculada pelas características físicas do material como módulo de Young e coeficiente de Poisson, $\boldsymbol{u}$ é o vetor de deslocamentos de cada elemento finito, e $\boldsymbol{f}$ é o vetor de forças atuantes em cada elemento.

Para que a estrutura final esteja em equilíbrio, desejamos obter o conjunto de deslocamentos dos nós na condição que as forças atuando em cada nó se cancelem. Esta condição gera um conjunto de equações com infinitas soluções, porém podemos resolvê-lo impondo condições de contorno, por exemplo regiões de deslocamento proibido ou permitido em apenas uma dimensão, como partes presas ou engastadas, deslocamentos iniciais e pontos em que forças externas serão aplicadas, e considerando que desejamos o resultado que minimiza a soma das tensões internas da estrutura.

3.3. Métodos Evolucionários: ESO, BESO e SIMP

Os métodos computacionais de resolução de problemas de otimização topológica foram descritos em 1988 por Bendsøe e Kikuchi [6], sua implementação tradicional são os programas de código aberto BESO e SIMP [7], que se utilizam largamente do método de elementos finitos. No problema de otimização estrutural, o objetivo é reduzir a massa de material utilizado em uma estrutura preservando maior rigidez mecânica. No algoritmo ESO, partindo da análise inicial da estrutura pelo método de elementos finitos, é obtida a distribuição de tensões internas do material, denominada de sensibilidade*,* e a somatória das sensibilidades é denominada *compliance,* cuja minimização é um dos meios de maximizar a rigidez estrutural.

Podemos observar um análogo à sensibilidade ao analisar estruturas plásticas sob luz polarizada, conforme mostrado na figura 3.3. Os pontos de acúmulo de estresse mecânico aparecem nos pontos de rápido gradiente de cores na figura, e correspondem no nosso problema à pontos de alta sensibilidade. Tais pontos tendem a apresentar rupturas mecânicas e deformações plásticas, por isso diminuir a sensibilidade local, e, portanto, global (*compliance),* equivale a aumentar a rigidez mecânica da estrutura. Neste trabalho a terminologia *sensibilidade* se refere exclusivamente à explicada anteriormente, não à sensibilidade como uma característica estatística, muitas vezes usada como métrica de redes neurais.

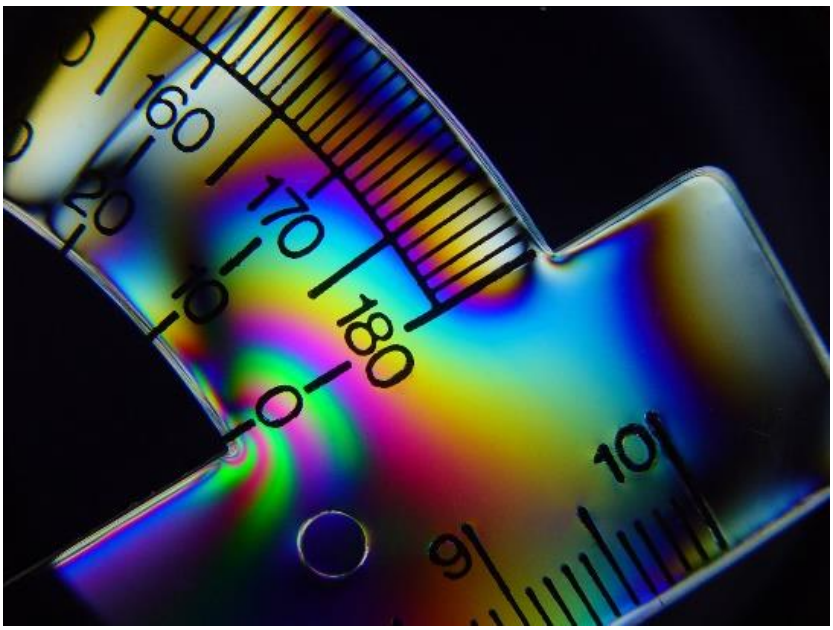

Figura 3.3: peça plástica sob luz polarizada, as regiões de alto gradiente de cores indicam alta concentração de tensões internas. Extraído de [8]

Nos métodos computacionais citados, são listados os elementos com menor sensibilidade, ou seja, elementos com baixas tensões e que tendem a ser subutilizados pela estrutura, portanto podem ser removidos com o menor comprometimento à rigidez mecânica. É feita então uma nova análise de elementos finitos, obtendo uma nova sensibilidade e removendo pontos conforme a lógica anterior, processo é repetido iterativamente até atingir uma desejada fração do material inicial, denominada fração de massa.

O método BESO (*Bidirectional Evolutionary Structural Optimization*), se apresenta como uma implementação mais sofisticada do algoritmo ESO, no qual elementos não só são removidos, como também podem ser adicionados nas proximidades de pontos de maior tensão. É imposto que o fator de adição deve ser menor que o de remoção para que a estrutura ainda convirja para a remoção desejada. É tido que tal flexibilidade gera melhores resultados com pouco custo computacional a mais, já que boa parte deste é dedicado ao cálculo da sensibilidade, refeito a cada passo do treinamento.

O método SIMP (*Simplified Isotropic Material with Penalization*), parte da mesma análise de elementos finitos que os descritos anteriormente, porém ao invés de adicionar ou remover elementos, estes podem ter densidades contínuas entre 0 e 1. O método consiste em iterativamente alterar essas densidades de modo a diminuir a densidade média até o valor de material desejado, minimizando sensibilidade. Os elementos podem ter densidades intermediárias contínuas dentro de um intervalo entre 0 e 1, representando respectivamente a ausência de elemento e a densidade máxima, porém são penalizados elementos com densidades intermediárias. A penalização de densidades intermediárias garante que boa parte da estrutura estrará na densidade máxima 1, e os vazios devidamente na densidade 0, sendo que os valores

intermediários tenderão a ocorrer apenas nas bordas do material enquanto este sofre as mudanças iterativas. Nos passos finais, essa penalização é aumentada de modo que a estrutura como um todo seja discretizada em densidades 0 ou 1.

Um exemplo deste processo é ilustrado nas figuras 3.4 e 3.5, conhecido como problema do *short-cantiléver*. A primeira figura é um diagrama da estrutura a ser otimizada, o domínio é a região de trabalho do otimizador, inicialmente preenchida de matéria em preto, a barra vermelha à esquerda representa um engaste em toda lateral esquerda, e a seta à direita uma força aplicada naquele ponto. A segunda figura mostra a evolução de tal estrutura ao ser otimizada pelo método SIMP, novamente as regiões escuras representam locais com maior densidade de massa, e as em branco os vazios.

Figura 3.4: Diagrama do problema do short-cantiléver, a barra à esquerda representa engastes em toda lateral esquerda, e a seta à direita um ponto de carga.

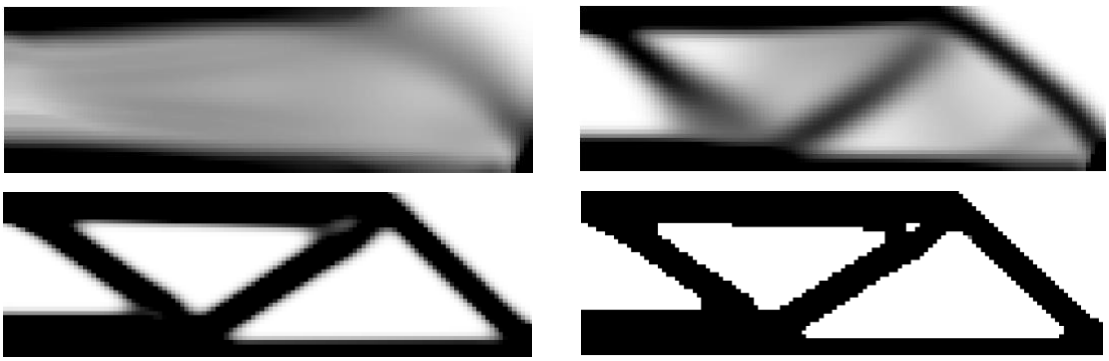

Figura 3.5: Evolução do processo de otimização estrutural pelo SIMP, mostrando em sequência usual de leitura a topologia nas iterações 3,13,30 e 80. Extraído de [1]

Tais métodos oferecem resultados efetivos para os problemas propostos, e são conceitualmente interessantes por se manterem atrelados a medidas e princípios físicos, como forças, deslocamentos e tensões de mola. Porém, são de alta demanda computacional, e podem se tornar proibitivos em problemas de alta resolução ou complexidade. A demanda crescente por métodos computacionalmente mais eficientes de abordar a otimização topológica levou

Sosnovik a propor em 2017 um método de criação e treinamento de redes neurais artificiais para esse fim.

3.4. Redes Neurais Artificiais

Enquanto uma explicação detalhada dos algoritmos e detalhes matemáticos de treinamento de Redes Neurais Artificiais (RNA) está fora do escopo deste trabalho e pode ser explorado melhor em [9,10], uma introdução ao assunto pode ser o suficiente para contextualizar seu funcionamento em termos de conceitos mais conhecidos, nos limitado ao tipo de rede e de treinamento utilizada nos trabalhos citados.

Em sua forma mais básica, uma RNA pode ser entendida como uma função matemática *f (entradas, parâmetros) = saídas*. Ao contrário de outras formas de programação, onde a função e seus parâmetros são conhecidos ou definidos explicitamente, em uma RNA sob treinamento dito supervisionado, apenas as entradas e saídas são conhecidas, e o processo de se obter os parâmetros é o treinamento da rede.

A estrutura básica de uma RNA para aprendizado supervisionado tem uma analogia rudimentar com o cérebro e aprendizado humano, e a utilização de termos neurônios e sinapses é comum na literatura. Uma entrada dada à rede é transmitida por diversas camadas de neurônios, cada um deles conectado a todos da camada anterior e da seguinte por sinapses. Neurônios recebem o sinal de todos os neurônios da camada anterior, aplicam uma função matemática não-linear (*activation function*) à soma destes sinais, acrescida de um peso próprio (*bias)*, e propagam o novo sinal para a camada seguinte, ao final a última camada resulta em uma saída desejada.

Tais analogias são computadas no programa como tensores e operações tensoriais. Matematicamente a rede se comporta como uma função que recebe uma entrada e parâmetros de pesos e *bias*, e que resulta em uma saída desejada. As entradas codificam informações externas para a primeira camada de neurônios através de valores numéricos, por exemplo uma imagem pode ser convertida em valores binários, 0 para branco e 1 para preto. Já as saídas assumem uma distribuição de probabilidades, no qual cada neurônio da camada final pode representar uma variável categórica, classe, ou conceito abstrato, como um dígito ou letra específica, ou mesmo conceitos *cão* e *gato*, a depender das especificações do problema e do *dataset.* O neurônio mais ativo será a predição da rede, e o valor associado a ele o grau de certeza.

Toda essa construção não se mostra imediatamente útil, pois a princípio os parâmetros ideais que fazem a rede produzir a predição correta para uma dada entrada não são conhecidos ou mesmo obtiveis por um humano em tempo exequível. Porém o apelo para o uso de RNA's se deve à técnica de *deep learning,* que utiliza dados de entradas e saídas previamente classificadas

por humanos ou obtidos por observações (aprendizado supervisionado), ou por simulações e demais métodos computacionais (aprendizado semi-supervisionado), denominados *datasets*, para o "treinamento" da rede. O processo de treinamento consiste em inicializar a rede com parâmetros aleatórios, apresentar uma entrada **X** e tomar a predição da rede **Y'**, inicialmente aleatória e errônea, e comparar com o resultado esperado **Y**, denominado *label*. O erro (*loss*) da rede para o exemplo é calculado através da *loss function* da forma *f(Y,Y')*, que em sua forma mais simples pode ser a diferença quadrática entre os vetores **Y** e **Y'**. Neste trabalho será amplamente usada a função custo chamada entropia binária cruzada [11], dada pela relação (2). Observa-se que a função tem mínimo em **Y** = **Y'**, e aumenta conforme quanto maior | **Y** – **Y'**|.

$$C = -\frac{1}{n}\sum_{1}^{n}[y \ln y' + (1-y)\ln(1-y')] \tag{2}$$

Um algoritmo de *backpropagation* ajusta levemente os parâmetros de rede **W**, de modo a minimizar a *loss.* Esse processo se repete com todos os exemplos de treino e por diversas vezes, cada uma denominada uma época, até chegar ao ponto em que os parâmetros **W** estão ajustados de maneira ótima, e quando exposta a qualquer entrada do *dataset*, a rede é capaz de prever corretamente a saída esperada.

A rede então pode ser extrapolada para casos não vistos no treinamento e ainda performar bem, a rede "aprendeu" através dos exemplos sem ser explicitamente programada, demonstrando sua capacidade de generalização. Para tal, é comum a prática de separar parte do *dataset* fora dos exemplos de treinamento, chamado de *dataset* de validação, reservados para a testar da performance da rede e sua capacidade de generalização. Apesar do processo de treinamento ser computacionalmente demandante, após realizado, a etapa de predição da rede para um exemplo é extremamente rápida, consistindo de poucos microssegundos, que é o principal atrativo para seu uso em problemas de otimização topológica.

Um comportamento indesejado comum de se observar em redes neurais é o *overfitting*, que ocorre quando a rede "decora" as respostas dos problemas de treinamento, porém sem capacidade de generalização. A rede pode sofrer overfitting por vários motivos, entre eles treino excessivo, número muito grande de variáveis treináveis, dataset pequeno, entre outras. Existem maneiras de se detectar a ocorrência de *overfitting*, como por exemplo quando o *loss* de treinamento continua caindo após certa época, mas o *loss* de validação começa a subir. Uma maneira de lidar com *overfitting* empregada neste trabalho será o uso de camadas de *Dropout* [12], cuja função é desativar um certo número de neurônios da camada anterior aleatoriamente a cada época, fazendo com que a rede não possa contar com nenhum neurônio ou caminho específico para resolução de um problema. Também faremos uso de *data augmentation*,

expandindo artificialmente o *dataset* através de rotações e simetrias, criando assim maior variabilidade no *dataset* sem o custo de gerar novos exemplos. sensibilidade

3.5. Redes Neurais Convolucionais, segmentação semântica e U-Net

As Redes Neurais Convolucionais (CNN) são uma das mais bem engenhosas aplicações das ideias principais das RNA [10], porém especializadas em trabalhos envolvendo entradas com relações espaciais, tal qual é o caso com imagens, devido à sua qualidade de invariância translacional, ou seja, reconhecer elementos independente da sua posição relativa na imagem. Tais redes são computacionalmente eficientes em tarefas como classificação de elementos, reconhecimento e transcrição de texto manuscrito, detecção de rostos e mesmo expressões faciais, identificação de espécies de plantas e animais. Redes CNN tem apresentado resultados consistentes aplicados à área da física computacional. Um exemplo de sucesso é o uso de CNN's na interpretação de diagramas de física de altas energias, como os obtidos em aceleradores de partículas. No artigo [13] é discutido uma variação quântica da CNN que é aplicada à classificação de neutrinos.

Nas CNN [14], uma camada convolucional é composta por filtros, também chamados de *kernels*, geralmente de tamanho 3x3 ou 5x5, que contêm valores numéricos e são passados sequencialmente por todos os pixels da rede, aplicando o produto interno dos seus valores com os dos pixels das imagens. O resultado dessa operação é chamado de *feature map*. Em uma rede CNN, os valores dos elementos dos filtros são os neurônios a serem treinados, que após o treino resultam em filtros capazes de extrair informação relevante da imagem. O processo da convolução está ilustrado na figura 3.6.

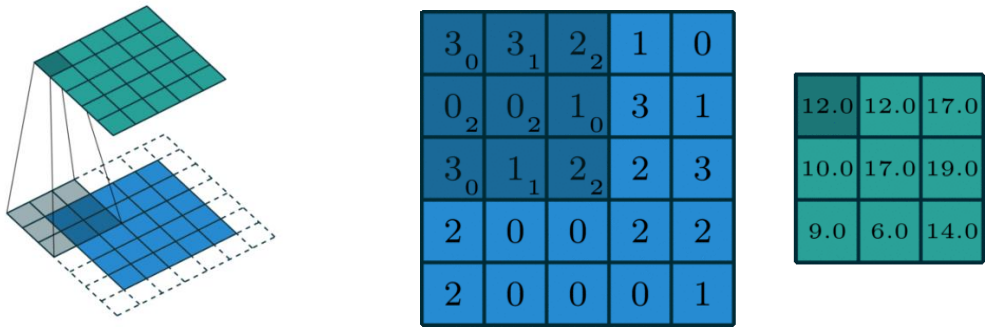

Figura 3.6: À esquerda, é representada a imagem em azul, o kernel *em cinza e o* feature map *gerado em verde. A operação de convolução consiste na "varredura" da imagem pixel a pixel. À direita é representada a operação matemática da convolução, onde é aplicado o produto interno do* kernel *com os valores dos pixels da imagem. Extraído de [12]*

Diversas camadas convolucionais podem ser aplicadas em sequência, as camadas seguintes sendo aplicadas nos *feature maps* das camadas anteriores. Quando bem treinadas, camadas iniciais codificam informações grosseiras das imagens, como linhas e bordas, e as finais informações mais abstratas, como formas, tamanhos e rotações. Um trabalho anterior apresentado por este aluno focou na exploração destas camadas e explicação didática do seu

funcionamento, e uma ilustração deste processo de extração da informação é apresentado na figura 3.7. Notamos que as imagens das últimas camadas não são facilmente interpretáveis, mas ao usar esta última camada como entrada para uma RNA, a rede pode converter imagens abstratas em previsões de classes corretas para uma imagem.

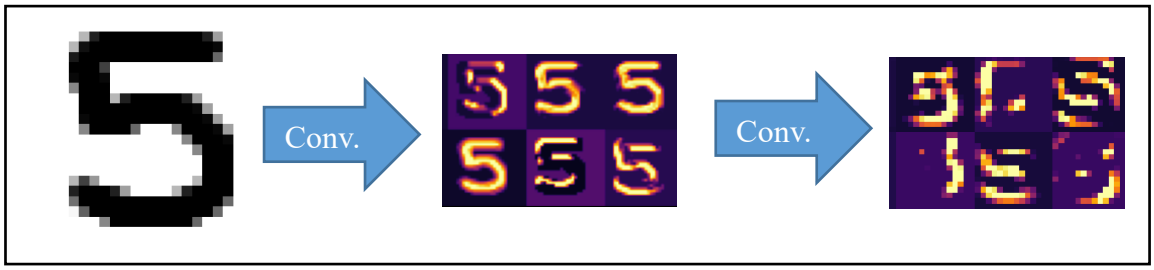

Figura 3.7: à esquerda: figura do algarismo "5" retirada do dataset MNIST. Ao centro e à direita: os feature maps *gerados após uma e duas camadas convolucionais, respectivamente.*

Os problemas atacados por redes neurais descritos até agora são agrupados em problemas ditos de classificação (prever qual classe) ou de regressão (prever um número). Um outro problema de suma importância prática é o da segmentação semântica: dada uma imagem, como classificar cada pixel desta em diversas classes, não apenas a imagem como um todo, ou seja, como segmentar pixel a pixel? A figura 3.8 mostra uma possível segmentação semântica de uma foto de um gatinho.

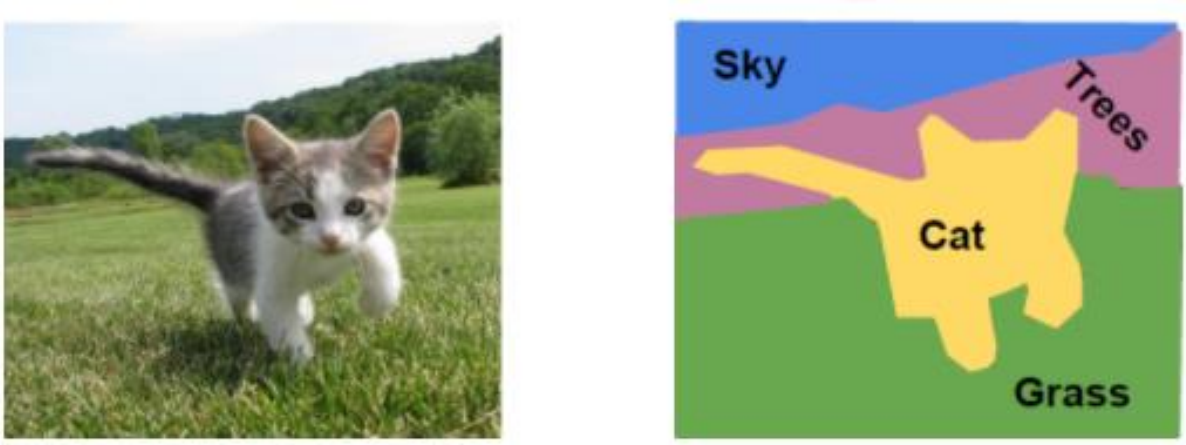

Figura 3.8: à esquerda uma fotografia de um gatinho em um campo, à direita uma possível segmentação semântica da imagem em 4 classes (de cima para baixo): "céu"," árvores", "gato" e "grama". Modificado de [15]

Uma proposta célebre para o problema de classificação semântica é a rede U-Net [16], que foi inicialmente aplicada na área de biomedicina em imagens de estruturas neuronais. A arquitetura de rede U-Net consiste em uma aplicação sequencial de camadas de convolução e convolução transposta (equivalente à operação inversa da convolução), intercalada com camadas de concatenação, que formam pares de camadas de convolução e convolução transposta de mesmo tamanho e número de filtros. Tal arquitetura consegue propagar informação abstrata de maneira eficiente pelo processo de convolução transposta, e as camadas de concatenação ajudam a manter a coerência espacial da previsão. O resultado é uma

arquitetura conceitualmente simples, de rápido treinamento e muito eficiente na tarefa de classificação semântica. A figura 3.9 mostra um diagrama dessa arquitetura, em que o evidente formato de "U" dá o nome à arquitetura. A metade da esquerda codifica informações da imagem, e é denominada *encoder*, já a metade direita decodifica informação abstrata de volta em uma imagem, e é chamada *decoder*. As camadas superiores codificam/decodificam informações ligadas à estrutura da imagem, enquanto às inferiores codificam informações abstratas.

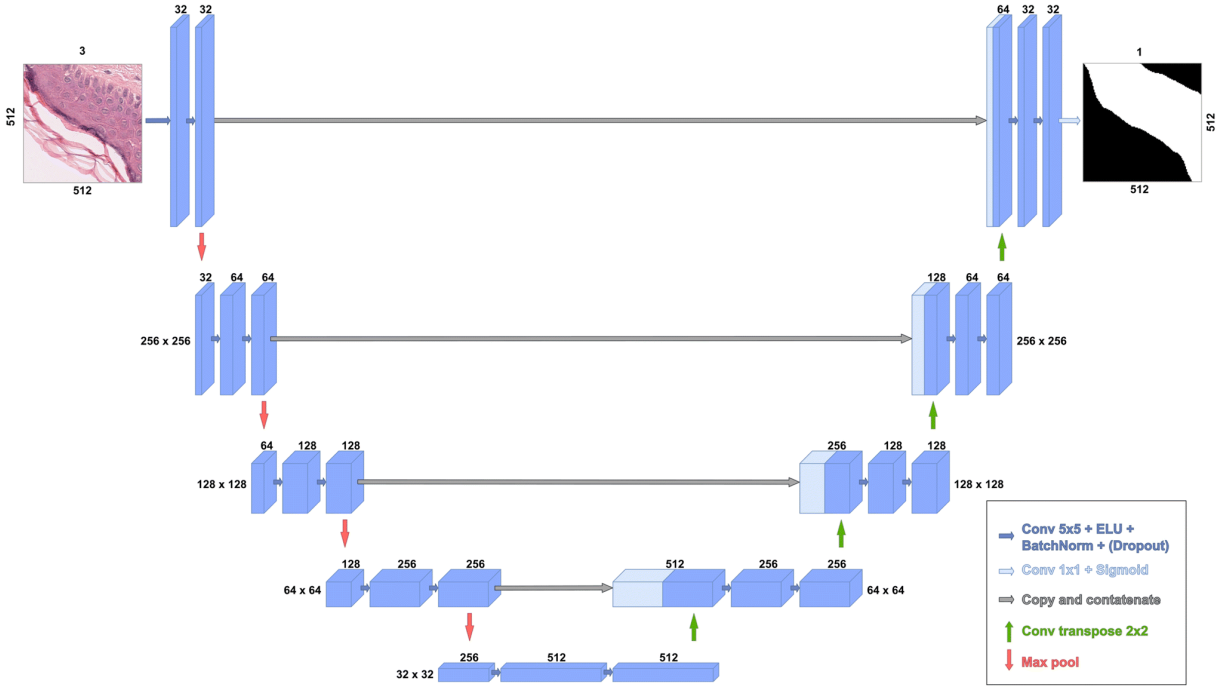

Figura 3.9: diagrama representando a arquitetura da rede U-Net. As camadas azuis representam as camadas convolucionais, com seu número de kernels *acima e as dimensões da camada ao lado. As setas para baixo representam operações de* max pooling*, enquanto as para cima de convolução transposta, já as cinzas conectam camadas de lados opostos da rede pela operação de concatenação. A imagem da esquerda é a entrada enquanto a da direita é a previsão da rede para a segmentação semântica em duas classes. Extraído de [17]*

3.6. Sosnovik e Oseledets – Otimização topológica como segmentação semântica

Em seu trabalho pioneiro de 2017, Sosnovik e Osedelets [1] propõem atacar o problema de otimização topológica como um de segmentação semântica, alavancando as já provadas capacidades da arquitetura U-Net. O presente trabalho foi muito inspirado no de Sosnovik, seguindo a mesma linha geral, arquitetura de rede e metodologia de implementação, buscando testar novidades conceituais e melhorar os resultados obtidos.

O método iterativo SIMP apresenta duas etapas bastante distintas: a primeira e relativamente curta em que a forma geral da estrutura é obtida de maneira ainda difusa; e a segunda de refino, significativamente mais custosa computacionalmente, em que a estrutura converge para sua forma final discreta. A segunda parte do problema pode então ser entendida como uma segmentação semântica, em que a partir de uma imagem inicial difusa, o programa pretendido deve classificar os pixels da imagem final em 0 ou 1, indicando ausência ou presença

do material. Tal papel pode ser performado com uma rede tipo U-Net, treinada previamente a partir de estruturas parcialmente otimizadas.

A rede proposta usa a arquitetura U-Net, e sua implementação é detalhada na figura 3.10. Como entrada, a rede toma uma imagem de dois canais: um da topologia parcialmente otimizada, e um representando o gradiente da otimização topológica com o último passo, que será explicado em frente. A saída da rede é uma imagem de mesma dimensão que as entradas, representando a estrutura predita.

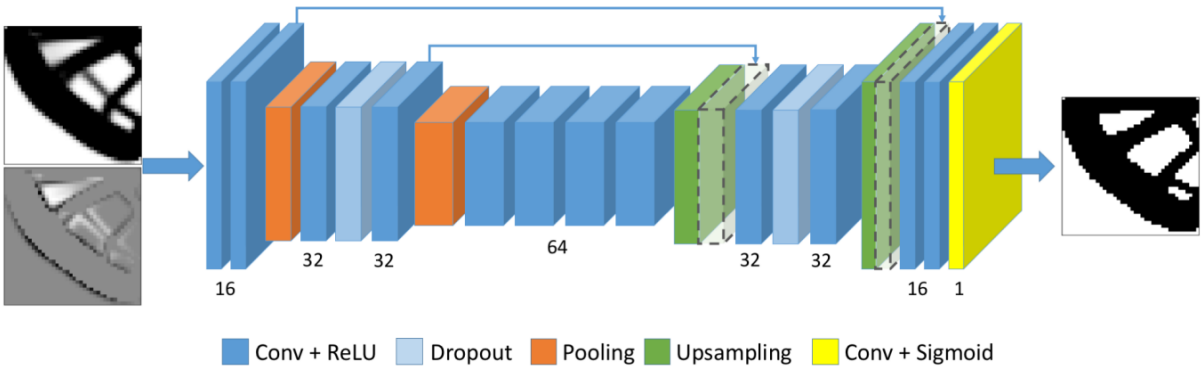

Figura 3.10: funcionamento da rede U-Net de Sosnovik, o número de kernels de cada camada é escrito abaixo da mesma, à esquerda as entradas são topologia e gradiente, e à direita a saída é a previsão da rede. Retirado de [1].

A elaboração do dataset consistiu na geração pelo programa SIMP de 10,000 exemplos de treinamento, cada uma de tamanho 40x40 e iterada 100 vezes. As posições dos pontos de engaste nas direções x e y, assim como dos pontos de força, foram sorteadas de modo a se concentrarem nas bordas da estrutura, e todas as forças definidas como unitárias apontando para -y, e a fração desejada de material sorteada de uma distribuição normal centrada em 0,5.

Foram então realizados diversos treinamentos independentes da rede em que foram escolhidos como entradas diversos pontos de "parada" para o otimizador, cujo índice é denominado pela variável *k*, através de distribuições de Poisson centradas em 5,10 e 30 e uma distribuição uniforme entre 1 e 100, denominadas respectivamente CNN P(5), P(10), P(30) e CNN U[0,100]. O gradiente pôde ser obtido diretamente pela diferença da imagem *k* com a *k-1*. A saída em todos os casos foi simplesmente a iteração final em *k=100*. Para cada treinamento o *dataset* também passou por um *data augmentation*, em que, a cada época, cada um dos exemplos é aleatoriamente trocado por suas simetrias do grupo D4 (rotação por 90º e espelhamentos nos eixos x e y). Este passo será explicado em mais detalhes na seção de metodologia.

A *loss function* escolhida consiste na soma de dois termos na forma da equação 3. O termo $\mathbf{L}_{\mathbf{conf}}$ corresponde à entropia binária cruzada, detalhada na seção 3.3. Já o termo aditivo $\mathbf{L}_{\mathbf{vol}}$, detalhado na equação 4, é dado pela diferença quadrada entre os valores de densidades médias

da predição e do *label* e pode ser modulado pelo hiperparâmetros **β**, sua função é convergir o resultado para a fração de volume desejada.

$$L = L_{conf}(y, y') + \beta L_{vol}(y, y') \tag{3}$$

$$L_{vol}(y, y') = (\langle y \rangle - \langle y' \rangle)^2 \tag{4}$$

As métricas usadas para medir a eficiência da rede são a acurácia binária (*binary accuracy)* e a *intersection over union* (IoU), seu uso é comum em diversos trabalhos no campo de segmentação semântica. A *binary accuracy* é simplesmente o número de pixels classificados corretamente sobre a o total, enquanto a métrica IoU é considerada mais precisa para problemas de muitas classes ou em que as classes são representadas desproporcionalmente, e são explicadas em detalhes em [18]. Idealmente uma boa rede apresentará ambas as métricas altas.

Os resultados do trabalho de Sosnovik e Osedelets estão sumarizados na tabela 3.1, e provam que a rede proposta é eficiente na resolução dos problemas de otimização topológica, abrindo um campo de pesquisa do qual frutificaram inúmeros trabalhos posteriores. Porém é válido destacar que os resultados listados foram obtidos sob o próprio *dataset* de treinamento, sem o emprego de um *dataset* de validação. Tal prática ainda era comum à época da publicação do trabalho, mas não permite observar se a rede está sofrendo de *overfitting*, e tende a resultar em métricas significativamente mais altas do que as que seriam caso a mesma rede fosse testada em dados de validação (não vistos por ela previamente).

	Iteration								
Method	5	10	15	20	30	40	50	60	80
Thresholding	92.9	95.4	96.5	97.1	97.7	98.1	98.4	98.6	98.9
CNN $P(5)$	**95.8**	97.3	97.7	97.9	98.2	98.4	98.5	98.6	98.7
CNN $P(10)$	95.4	**97.6**	**98.1**	98.4	98.7	98.9	99.0	99.0	99.0
CNN $P(30)$	92.7	96.3	97.8	**98.5**	**99.0**	**99.2**	**99.4**	**99.5**	99.6
CNN $U[1, 100]$	94.7	96.8	97.7	98.2	98.7	99.0	99.3	99.4	**99.6**

	Iteration								
Method	5	10	15	20	30	40	50	60	80
Thresholding	86.8	91.2	93.3	94.3	95.6	96.3	96.8	97.3	97.9
CNN $P(5)$	**92.0**	94.7	95.4	96.0	96.5	96.9	97.1	97.3	97.4
CNN $P(10)$	91.1	**95.3**	**96.4**	96.9	97.4	97.8	98.0	98.0	98.1
CNN $P(30)$	86.4	92.9	95.7	**97.0**	**98.1**	**98.5**	**98.8**	**99.0**	99.2
CNN $U[1, 100]$	90.0	93.9	95.5	96.4	97.5	98.1	98.6	98.8	**99.2**

Tabela 3.1: resultados obtidos pelas redes de Sosnovik. Acima para acurácia binária, e abaixo para IoU. Foram testadas 4 redes, descritas anteriormente, enquanto a descrita como "thresholding" atua como controle. Extraído de [1]

3.7. Proposta de Neves e Pavanello – Uso de sensibilidades como entrada

O modelo de rede proposto por Neves e Pavanello [2], ambos parte do LTM- Laboratory of Topology Optimization and Multiphysics Analysis, vinculado à UNICAMP, foi inspirado por Sosnovik, porém com modificações conceituais relevantes. A começar pelo problema atacado, o trabalho de Neves optou pelo *short cantiléver*, um caso particular de problemas em que toda a borda esquerda da estrutura está engastada, e apenas uma força de valor constante é aplicada à sua borda direita, em uma altura diferente à cada exemplo. Utilizando o método BESO (ao invés do SIMP de Sosnovik), foi gerado um *dataset* de estruturas parcialmente otimizadas de apenas 48 imagens de 128 pixels de lado cada, correspondendo a pontos de aplicação da força distribuídos uniformemente, sendo que apenas 24 delas foram usadas no treinamento e o restante reservado para validação, um número ínfimo comparado aos 10.000 exemplos de Sosnovik.

Uma diferença crucial da rede de Neves foi o uso de apenas um canal como entrada: a sensibilidade condicionada, em oposição ao emprego de dois canais de Sosnovik: a topologia e gradiente. Definida a fração de massa desejada em 50%, a sensibilidade foi obtida interrompendo a otimização do BESO em diversos passos do iterador (equivalente aos diversos *k* do trabalho de Sosnovik), por exemplo quando a fração de massa chegava a 95,25%. Para condicionar a sensibilidade, os valores máximos de sensibilidade foram limitados à 1/30 do maior valor, e em seguida normalizados.

A função custo utilizada foi apenas a entropia cruzada binária, excluído o termo aditivo de volume $\mathbf{L}_{\mathbf{vol}}$ (eq. 4), permitindo que a rede focasse na minimização da sensibilidade. A saída da rede é a topologia final, porém esta é dada como uma distribuição de densidades. Para discretizar a estrutura e ao mesmo tempo garantir que a fração de volume desejada seja obtida, é aplicado um filtro binário à saída, em que densidades abaixo da mediana das densidades são definidas como 0 e as acima como 1. A figura 3.11 mostra um diagrama da rede aplicada.

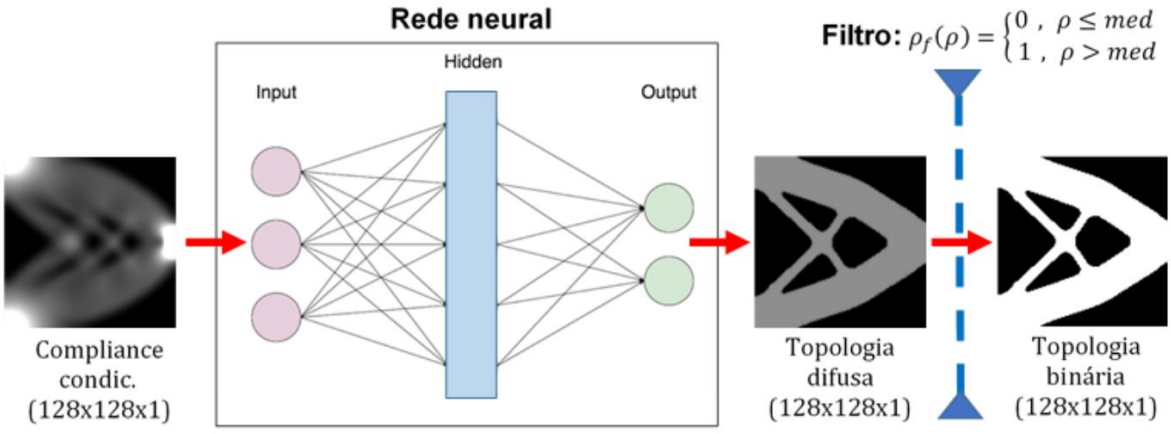

Figura 3.11: diagrama do funcionamento da rede U-Net de Neves. À esquerda a entrada da rede é a sensibilidade condicionada, que resulta em uma previsão de topologia difusa, que é a seguir passada pelo filtro binário, resultado em uma previsão de valores discretos para a topologia. Extraído de [2]

4. METODOLOGIA

4.1. Elaboração do *Dataset*

A fim de comparar diretamente as performances das redes propostas e testar novas hipóteses e variações, foi criado um *dataset* de 10.000 exemplos segundo as especificações descritas em [1]. Os dados foram gerados usando a biblioteca TOPY, uma implementação do método BESO na linguagem Python. Buscando a obtenção de um *dataset* mais genérico e versátil possível, foi incluído, para cada um dos exemplos, todos os passos de sua otimização, e a cada passo tanto a topologia quanto a sensibilidade, assim como metadados sobre o exemplo, como a localização dos pontos de engaste e força. Um exemplo da otimização é mostrado na figura 4.1.

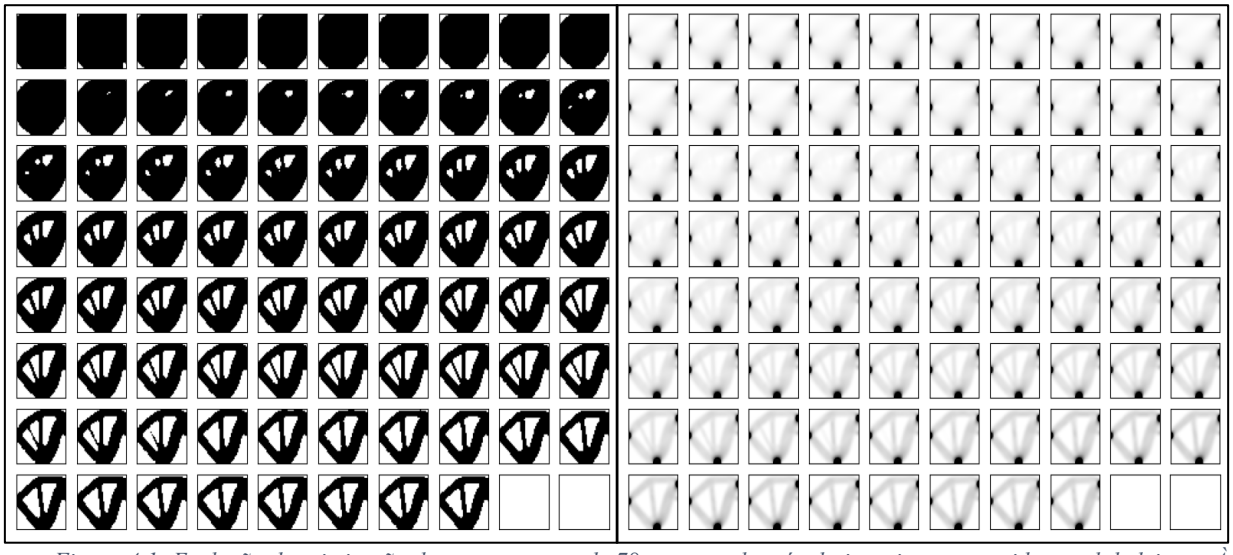

Figura 4.1: Evolução da otimização de uma estrutura de 78 passos pelo método iterativo, em sentido usual de leitura. À esquerda a evolução da topologia, e à direita da sensibilidade.

Para cada exemplo, o número de cargas (N_L) e de pontos de engaste em y (N_y) foram sorteados de distribuições de Poisson centrada em 1 e o número de pontos de engaste em x (N_x) segundo a distribuição de Poisson centrada em 2, mostradas na figura 4.2.

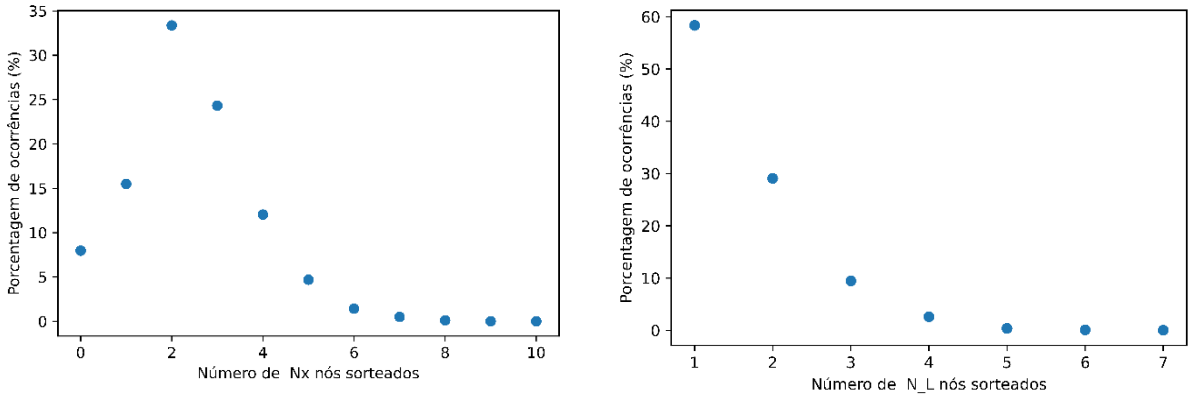

Figura 4.2: Gráficos representando as probabilidades de sorteio de N_x e N_L, seguindo distribuições de Poisson centradas em 2 e 1 respectivamente.

A força aplicada nos pontos de carga foi definida como 1N apontando para baixo. Foi então sorteada a localização de cada carga e engaste, de modo que fosse 100 vezes mais provável uma localização na borda ser sorteada em relação ao interior. Um *heatmap* das distribuições de carga pelo espaço de trabalho é mostrado na figura 4.3.

Figura 4.3: heatmap *das posições sorteadas para os pontos de carga (N_L), onde a grande maioria se concentra nas bordas da estrutura.*

O comportamento padrão do otimizador TOPY é diminuir quase linearmente a fração de volume a cada iteração até chegar na fração desejada, então continuar movendo massa pelo espaço de trabalho a fim de diminuir a *compliance*. Ao contrário *dataset* de Sosnovik que fixava o número de passos da otimização em 100 iterações, no nosso foi fixado apenas que o processo seria interrompido caso cumpridas duas condições:

- Atingida fração de volume desejada, definida em 50%.
- Não houvesse diminuição significativa da *compliance* após algumas iterações.

Uma curva típica da otimização topológica segue as linhas da figura 4.4, onde o passo do otimizador é definido pela variável *k.* Escolhemos então alguns *k* de interesse para seccionar o *dataset*, relacionados na tabela 4.1 com suas respectivas frações de massa.

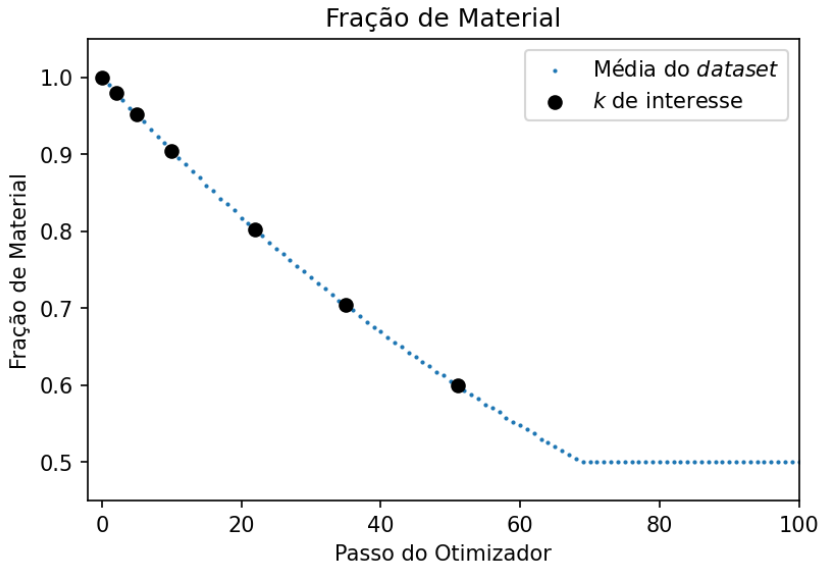

Figura 4.4: Fração de massa média pelo passo do otimizador. A massa é removida de maneira quase linear por passo, até que obtida a fração de massa desejada em 0.5, nos passos seguintes não há remoção apenas remanejo de massa. Os pontos pretos representam k *de interesse nos quais o* sub-dataset *é seccionado e a rede será testada.*

k	0	2	5	10	22	35	51
Fração de massa	1,0000	0,9800	0,9525	0,9050	0,8025	0,7050	0,6000

Tabela 4.1: a linha superior mostra os k *escolhidos para seccionar o* dataset*, e a inferior a fração de massa média correspondente.*

A separação do *dataset* principal em *sub-datasets* definidos pelo passo *k* serviu aos propósitos práticos tanto de facilitar as operações de condicionamento da sensibilidade, quanto de diminuir a memória RAM necessária para guardar os dados durante o treinamento de rede. Cada *sub-dataset* consiste de um conjunto de:

- entradas: sensibilidade do passo *k*
- *Labels*: topologia do passo final

Observamos que, a cada exemplo, os valores das sensibilidade*s* variavam entre 10^{-5} e 10^{-15}, sendo que os poucos maiores valores tendiam a se encontrar apenas próximos aos pontos de engaste e cargas, e a esmagadora maioria eram ordens de grandeza inferiores a estes. Como as redes convolucionais apresentam melhores resultados com valores normalizados, foram propostas diversas maneiras de realizar a normalização e condicionamento dos dados de modo a atender essa exigência. Após diversos testes o método escolhido foi uma normalização com *clipagem*, de modo que para cada exemplo, o valor do *clip* foi definido como cinco vezes a mediana das sensibilidade*s*. Valores acima do clip foram definidos em 1 e os abaixo divididos pelo *clip*. Foi também obtido um segundo conjunto de *sub-datasets* consistindo do gradiente espacial das sensibilidade*s*. A figura 4.5 mostra a comparação do mesmo exemplo apenas normalizado e com o condicionamento proposto. Já a figura 4.6 mostra a comparação de dois histogramas com a distribuição dos valores de sensibilidade.

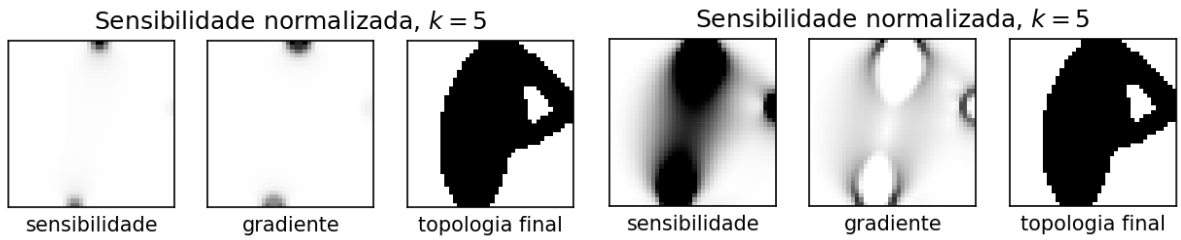

Figura 4.5: Representação das sensibilidades, gradientes e topologias finais de uma mesma estrutura, à esquerda obtida a partir da normalização, e à direita a partir da normalização mais clipagem descritas. As regiões pretas representam maiores valores.

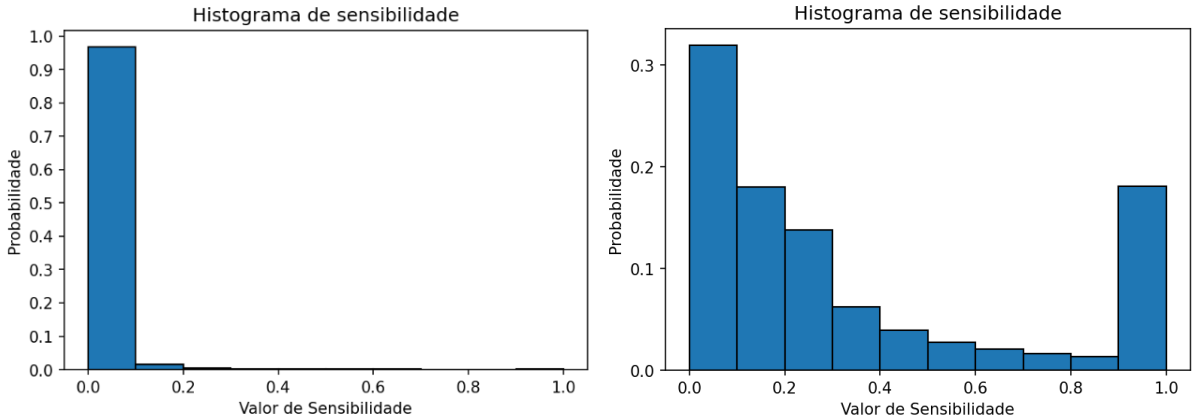

Figura 4.6: histogramas dos valores de sensibilidade do dataset*, à esquerda obtida a partir da normalização, e à direita a partir da normalização mais clipagem descritas, onde observa-se uma distribuição de valores mais regular no intervalo para o* dataset *normalizado partir do condicionamento proposto.*

4.2. Definição da Rede

O modelo de rede foi elaborado baseado na arquitetura U-Net para a linguagem Python e implementada em TensorFlow [19], seguindo a forma geral e configurações da rede proposta por Sosnovik, e hospedada na plataforma Google Colaboratory. Esta plataforma tem como foco desenvolvimento científico e aprendizado em *deep learning,* e apresenta a vantagem de executar o programa remotamente e com grande capacidade computacional, dispondo de placas de vídeo dedicadas de maneira gratuita, dispensando a necessidade de comprar máquinas próprias para tal, ou operar em computadores subcapacitados para a tarefa em questão. O código é implementado para rodar em placa gráfica (GPU) disponibilizada pela plataforma, uma NVIDIA Tesla K80, e uma medição simples revelou que o uso da GPU, em comparação com de CPU, acelera o processo de treinamento da rede em cerca de 24 vezes.

A arquitetura geral da rede está representada na figura 4.7, o otimizador escolhido foi o ADAM [20] com parâmetros *learning rate* = 0,001 e *épsilon* = 10^{-5}, o tamanho do *mini-batch* = 100 e o número de épocas de treino definido em 100. Também foi escolhido um valor de *drop_rate* = 0,1 para controlar a intensidade das camadas de *dropout.* A *loss function* escolhida foi a entropia binária cruzada, sem termos aditivos, e a função de ativação das camadas convolucionais a função “ReLU” [21].

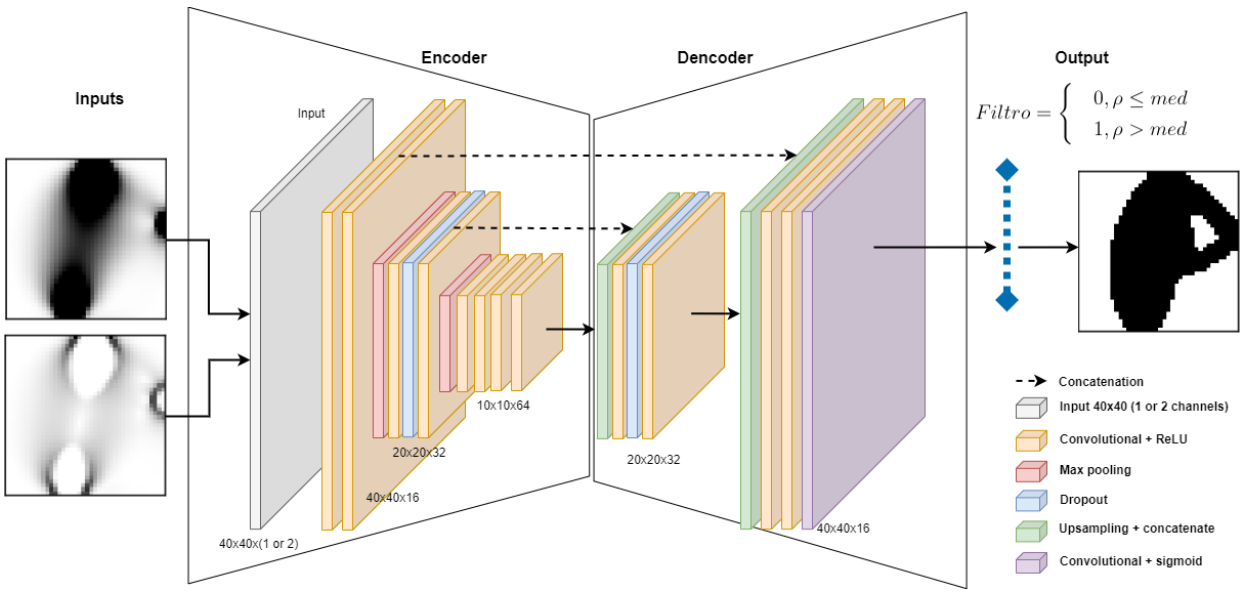

Figura 4.7: funcionamento da rede U-Net implementada. Cada retângulo corresponde a uma camada, cujas dimensões são dispostas abaixo do mesmo, sendo os dois primeiros números as dimensões da figura e o último o número de kernels. *À esquerda, as imagens representam as possíveis entradas, que são uma imagem de um ou dois canais: a topologia e/ou o gradiente espacial da mesma, e à direita a saída é a topologia prevista. As setas tracejadas indicam uma ligação de concatenação entre as camadas de mesma dimensão.*

Foi definida uma divisão aleatória do *sub-dataset* em *sets* de treino e de validação, sendo 8.000 exemplos para o *set* de treinamento e 2.000 para o *set* de validação. Foi também implementado um algoritmo de *data augmentation* na rede, explorando as simetrias dos problemas sob transformações do grupo D4. Esta operação tem como objetivo aumentar a variabilidade dos dados vistos pela rede durante o treinamento sem a custa computacional de gerar mais exemplos. A cada época de treino, todos os exemplos do *sub-dataset* passavam por três operações de simetria, cada uma com 50% de chance de ser aplicada independentemente, conforme exemplifica a figura 4.8.

- Espelhamento vertical
- Espelhamento horizontal
- Rotação anti-horária em 90 graus

Figura 4.8: exemplo de uma imagem do dataset *transformada sob as simetrias do grupo D4.*

Essa operação dá possibilidade de 8 simetrias por exemplo, efetivamente a rede tinha acesso à 64.000 exemplos para o treinamento, em que na média uma simetria específica seria vista

12,5 vezes. Deste modo aumentamos virtualmente o número de exemplos que a rede tem acesso para treino em 8 vezes, com maior variabilidade espacial. Ao final de cada época também a ordem dos exemplos é embaralhada aleatoriamente, para melhorar o *overfitting*.

A rede final possui cerca de 200 mil parâmetros, e foi treinada para cada k de interesse usando como entrada:

- 1 canal: sensibilidade condicionada.
- 1 canal: gradiente da sensibilidade.
- 2 canais: sensibilidade condicionada mais gradiente da sensibilidade condicionada.

Devido à natureza estocástica dos treinamentos, cada um foi repetido por no mínimo 3 vezes de modo a diminuir flutuações aleatórias. As métricas de treinamento, performance final e modelos de rede treinada foram salvos para futura consulta e uso.

Todas as bibliotecas adicionais são de código aberto e foram implementadas em Python: os gráficos e figuras das imagens relevantes foram gerados usando a biblioteca aberta MatPlotLib [22], o condicionamento e manipulação dos dados foi feito através da biblioteca NumPy [23], e a visualização dos resultados através da biblioteca Pandas [24].

5. RESULTADOS

5.1. Métricas do *dataset*

Algumas métricas podem ser obtidas a partir do *dataset* gerado pelo método exposto. Uma métrica relevante é quantos passos do método iterativo são necessários para otimizar uma estrutura, pois aproximando que o otimizador gasta o mesmo tempo em cada passo (podemos fazer essa aproximação pela constatação que a cada passo, o maior tempo computacional é gasto no cálculo de elementos finitos, que independe do k), a fração de trabalho que será feita pela rede depende tanto do k escolhido, quanto da média de passos necessários para concluir a otimização pelo método tradicional (k menores então correspondem à maior trabalho da rede). No caso, em média a otimização foi completa em 86.9 passos. A figura 5.1 mostra um histograma do número de passos necessário para completar a otimização.

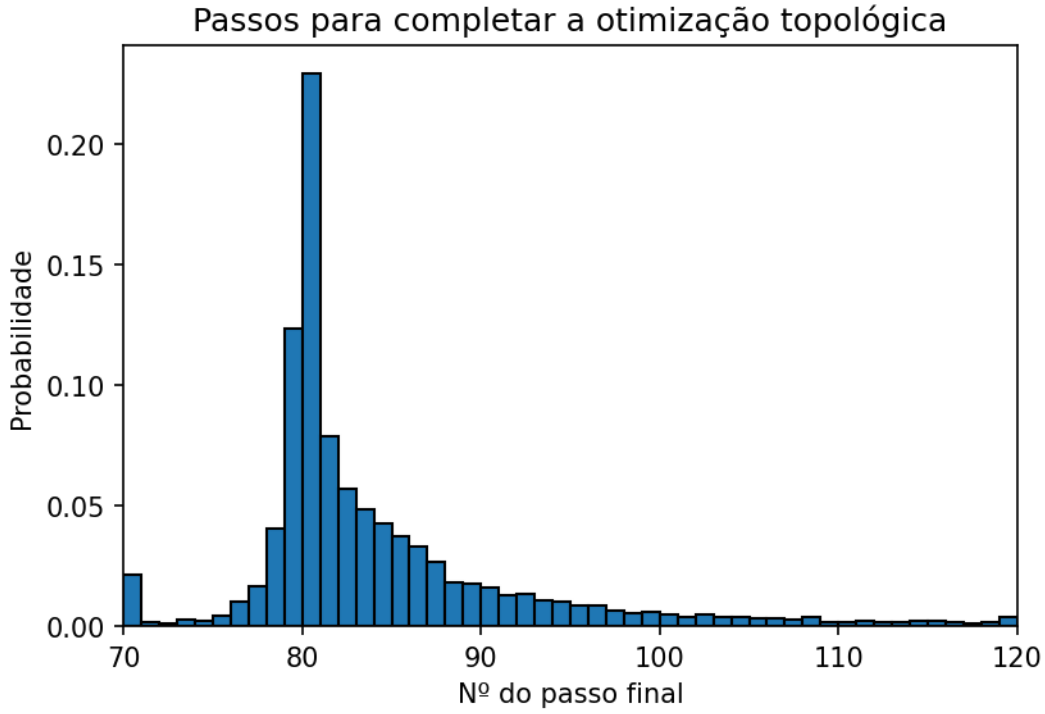

Figura 5.1: histograma do número de passos necessários para completar a otimização topológica, com média em 86.9 passos.

Podemos então calcular quanto do trabalho computacional estaria sendo performado pela rede, sumarizado na tabela 5.1:

k	0	2	5	10	22	35	51
Fração de trabalho	1,0000	0,9770	0,9424	0,8849	0,7468	0,5972	0,4131

Tabela 5.1: na linha superior o k escolhido para fracionamento do sub-dataset*, e na linha inferior a fração de trabalho correspondente para finalização da otimização topológica, considerando que todos os passos são de igual trabalho computacional.*

5.2. Resultados de treinamentos e predições

A figura 5.2 mostra um típico treinamento da rede, usando como entrada a sensibilidade condicionada em k=5. Os gráficos representam as métricas de *loss* e acurácia binária por época.

Em cada um, a curva em azul representa o resultado para o *dataset* de treinamento e a em laranja para o *dataset* de validação. A forma das curvas nos permite afirmar que não está ocorrendo *overfitting*, pois não existe um evidente aumento da *loss* de validação ao final do treino, e a suavidade da curva de treinamento, junto com ausência de fortes picos em qualquer curva, indica que os hiperparâmetros de *learning_step* e *batch_size* estão adequados.

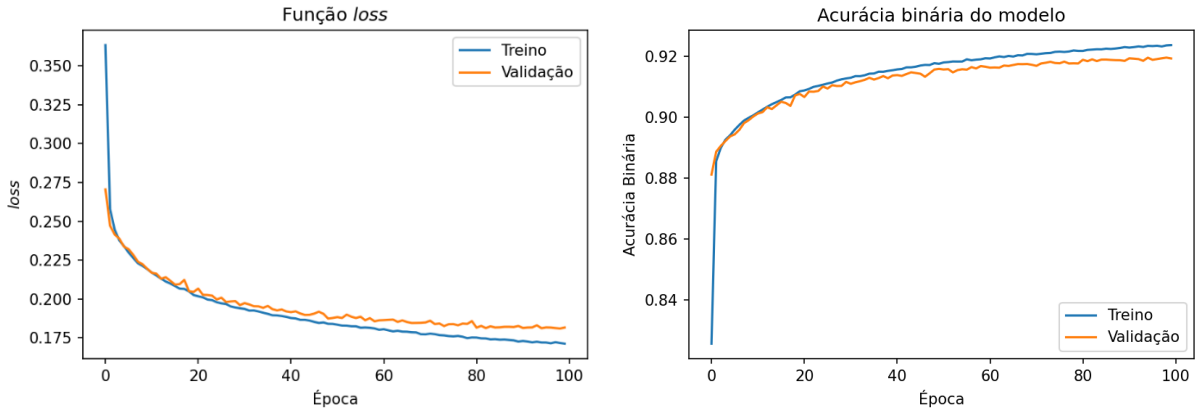

Figura 5.2: exemplo das curvas de treinamento para um treino de rede com k=5, para os dados de treinamento e validação, em azul e laranja respectivamente. À esquerda a função loss e à direita a acurácia binária.

As figuras 5.3 e 5.4 mostram o resultado de acurácia binária e IoU para diversos treinamentos para os *k* de interesse, expressos como fração de massa remanescente, o programa automaticamente guarda a configuração de rede com a melhor performance nos dados de validação. Cada treinamento foi repetido por 3 vezes e obtidas as médias e desvios. Foram omitidas as barras de erro, pois estas não seriam visíveis nos gráficos, os erros não superaram 0,5% dos valores de acurácia binária e IoU.

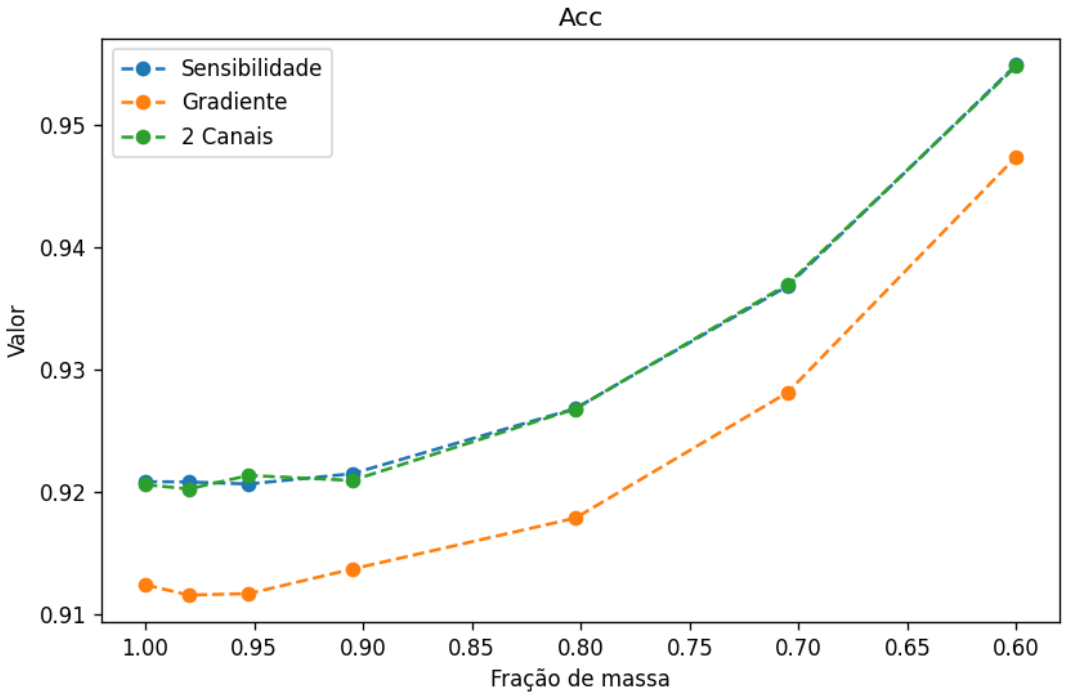

Figura 5.3: resultado de médias de acurácia binária por fração de massa, para treinamentos com k = {0,2,5,10,22,35,51} convertidos para suas respectivas frações de massa.

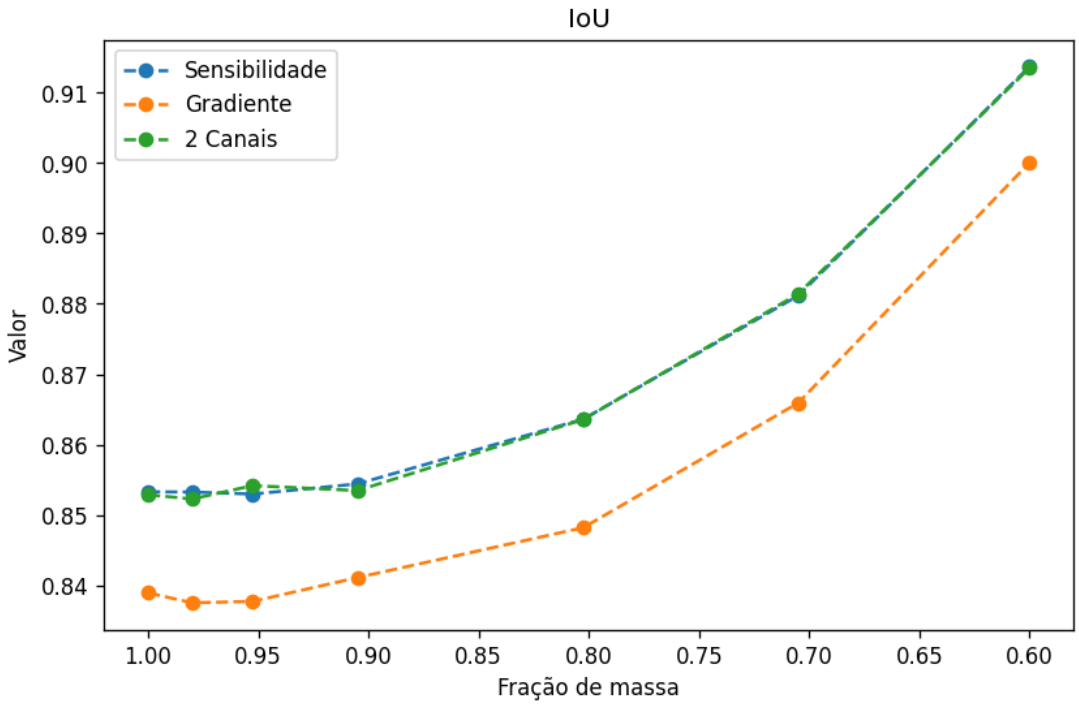

Figura 5.4: resultado de médias de IoU por fração de massa, para treinamentos com k = {0,2,5,10,22,35,51} convertidos para suas respectivas frações de massa.

Observamos que a rede proposta tem uma boa performance, acertando pelo menos 91% dos pixels no *dataset* de validação, e que também mantém sua boa performance mesmo quando k=0. Isto é uma evolução conceitual importante, e que será abordada à frente. Não é visto uma melhora significativa do uso de dois canais como entrada da rede em comparação com o uso apenas da sensibilidade, nem do uso apenas do gradiente espacial.

Em comparação com os resultados das métricas obtidas por Sosnovik, nos concentramos em comparar as performances das redes com k pequenos, isso significa uma situação em que houve ainda pouca otimização pelo método evolutivo e boa parte do trabalho será performado pela rede. Tomando uma comparação em k=5, a rede de Sosnovik obteve a acurácia binária de 95,8% e IoU de 92,0% de IoU, enquanto a rede proposta obteve 92,1% de acurácia binária e 85,3% de IoU. Porém é relevante lembrarmos que a rede proposta foi testada com dados de validação não vistos durante o treinamento, enquanto à de Sosnovik foi testada nos próprios dados de treinamento, recebendo assim uma vantagem.

O tempo gasto para treinar nossa rede, para cada k, foi cerca de 5 a 6 minutos, contra 80 a 90 minutos para as de Sosnovik. O número de parâmetros treináveis foi de 191.969 contra 192.113, evidenciando que ambas as redes estão muito próximas em tamanho e complexidade. Após treinada, a rede pode otimizar 10.000 exemplos em 1,59 segundos em média, independente de qual k era escolhido, enquanto os exemplos gerados pelo otimizador topológico demandavam cerca de 20 segundos cada, logo as redes eram cerca de 6 ordens de grandeza mais rápidas. Neste quesito, é necessário ressaltar que ambos os processos não foram

executados em paridade de condições, levando à uma vantagem para a rede. A otimização topológica através do BESO foi executada no programa MatLab, em uma máquina comum e rodando na CPU, enquanto a previsão da rede foi feita em Python através de GPUs, logo uma implementação otimizada do programa BESO poderia razoavelmente ser 2 a 4 ordens de grandeza mais rápida que a utilizada.

Após treinada, a rede é capaz de fazer previsões em exemplos não vistos anteriormente, a figura 5.6 mostra alguns resultados positivos em que as previsões são indistinguíveis da estrutura otimizada iterativamente.

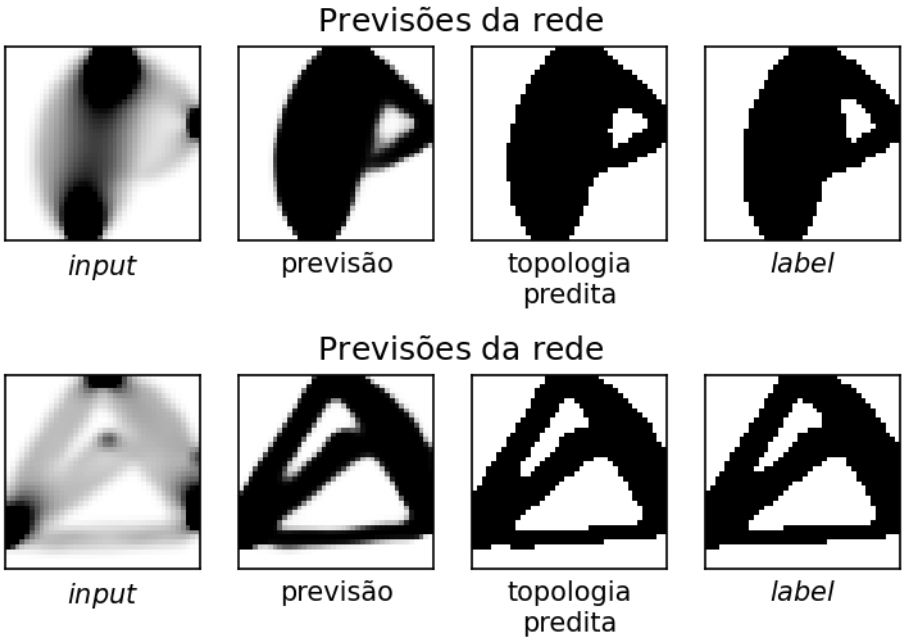

Figura 5.6: Resultados de treinamentos para sensibilidade em k=5, as topologias preditas são muito próximas das obtidas pelo método iterativo.

Em outros casos, a rede foi incapaz de generalizar o resultado, produzindo previsões significativamente diferentes dos *labels,* como mostrado na figura 5.7. Neste observamos fenômenos interessantes para serem abordados em trabalhos futuros, o principal é a presença de pedaços de estrutura desconexos, flutuantes, ou que evidentemente não servem para nenhum propósito estrutural, chamados de frustrações geométricas. Tais artefatos podem ser entendidos pelo fato de que a rede não foi ensinada nada sobre os atributos físicos do problema, conhecendo apenas imagens de treino. Entretanto, caso seja possível quantificar matematicamente tais frustrações geométricas, a princípio é possível penalizar a rede por produzi-las através da soma com um termo aditivo na função *loss*, melhorando a robustez da rede em problemas novos.

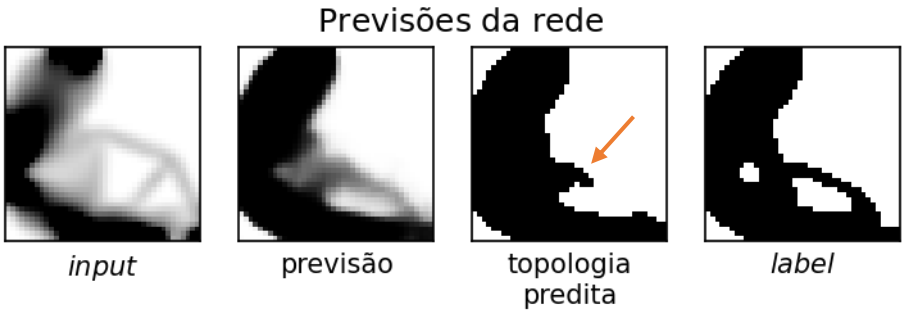

Figura 5.7: Resultados de treinamentos para sensibilidade em k=5, as topologias preditas são diferentes das obtidas pelo método iterativo. Observa-se em destaque indicado pela seta laranja uma frustração geométrica.

Também foi levantada a hipótese que tais frustrações geométricas são causadas como consequência da escolha do filtro binário, que pareceu se confirmar pela observação da ausência de frustrações geométricas nas estruturas difusas preditas pela rede, e que estas se evidenciam apenas depois da aplicação do filtro. Uma possível solução então seria a implantação de um filtro anterior ao filtro binário no qual seria aplicada uma difusão reversa, de modo que as densidades difusas tendessem a se concentrar antes do corte binário.

5.3. Treinamento com exemplos não-otimizados

Focando na utilidade da rede proposta como uma ferramenta aceleradora do processo de otimização topológica de estruturas, testamos até que ponto poderíamos "parar" o otimizador e ainda obter bons resultados com a rede. A princípio supomos que ao diminuir o número de passos do otimizador k, melhoraríamos a velocidade do processo de modo geral, mas perderíamos na acurácia das previsões de rede. Surpreendentemente para o autor, a rede performou bem até mesmo na situação em que o otimizador foi parado no primeiro passo, com k=0 e cerca de 100% da fração de massa. Em um treino nessas condições, a rede obteve acurácia binária = 92,1% e IoU = 85,3%. A rede pode fazer boas previsões partindo apenas da sensibilidade obtida através do método de elementos finitos, dispensando qualquer passo de otimização topológica.

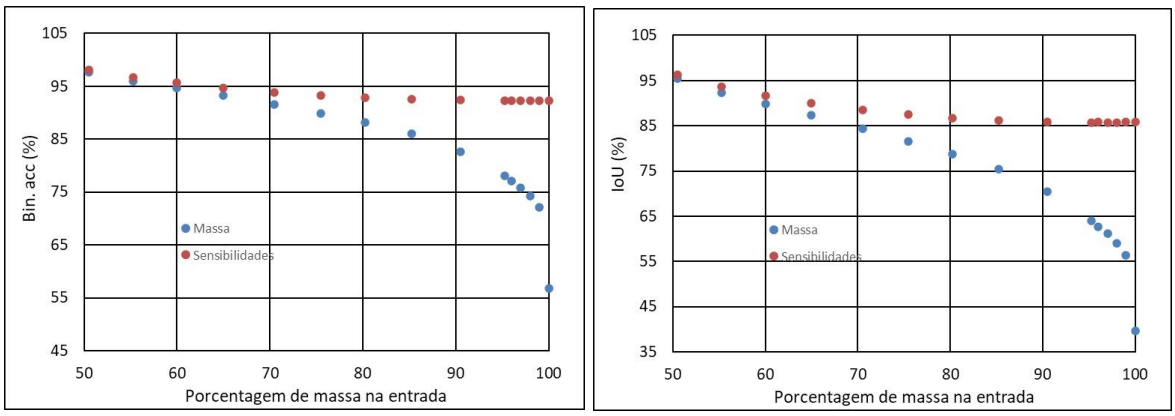

Figura 5.7: curvas de métricas obtidas por Neves, para entradas de topologia em azul e sensibilidade em vermelho, pela fração de massa. À esquerda a acurácia binária, e à direita o IoU. Observa-se melhor performance da entrada sensibilidade, principalmente à maiores frações de massa, correspondendo à menores k.

Levantamos a hipótese que a sensibilidade condicionada, mesmo no primeiro passo, já carrega toda informação necessária para a previsão acurada da estrutura. Isso está em evidente contraste com as limitações da rede de Sosnovik, que por receber como entrada a topologia e gradiente da topologia, não poderia prever a topologia final para *k=0*, pois em seu dataset a topologia de todos os exemplos é igual, 100% da massa uniformemente distribuída, e não existe ainda passo anterior com o qual se possa calcular o gradiente (definido como a diferença entre o passo *k* e *k-1,* porém *k-1* não é definido em *k=0*). De fato, na figura 5.7, uma comparação direta pode ser feita na nossa rede, usando como entrada a topologia e a sensibilidade de um mesmo *k.* Observamos que mesmo para altas porcentagens de massa de entrada (correspondendo a menores *k*), a performance da rede se estabilizou quando treinada com sensibilidades, mas caiu fortemente quando treinada com topologias, sendo que com *k=0* a acurácia binária se aproxima de 50%, evidenciando que previsão da rede treinada apenas com topologia é pouco melhor que um "chute" binário.

5.4. Perspectivas futuras

A conclusão deste trabalho deixa aberta muitas portas de pesquisa interessantes no campo do uso de *machine learning* para otimização topológica. Será objetivo de trabalhos em um futuro próximo estudar se a rede será capaz de fazer boas previsões recebendo como entrada não mapas de sensibilidade e topologia, mas sim apenas a descrição matemática do problema, com os pontos de engastes e cargas descritos claramente, não recorrendo nem ao passo zero do otimizador topológico.

O uso de arquiteturas diferentes também pode trazer resultados inovadores, por exemplo com GAN's, nas quais duas redes "competem", uma tentando gerar resultados "artificiais" cada vez melhores, e outra cujo objetivo é discernir entre os resultados "artificiais" e "reais", assim melhorando a performance de ambas. Também cogitamos o uso de redes de *reinforcement learning*, nas quais não são fornecidos *labels*, e a rede aprende através de resultados positivos e negativos, construindo um aprendizado cada vez mais complexo.

Modificações no filtro binário usado também podem trazer benefícios, inclusive na diminuição das frustrações geométricas. Nesse sentido a proposição de um filtro de difusão reversa como passo anterior ao filtro binário se mostra uma solução se simples implementação e grande potencial.

Por fim, é tentador imaginar que poderíamos usar uma previsão da rede treinada como ponto de partida para um otimizador topológico, desta maneira deixando o "trabalho grosso" dos passos iniciais para a rede e apenas o refino na estrutura final para o otimizador topológico. De fato, experimentos realizados por Neves nessa área já mostram resultados promissores, em que um exemplo com sensibilidade não otimizada (fração de massa = 100%, nenhum passo do

iterador topológico), foi passada por uma rede, e a topologia prevista então alimentada como entrada ao programa BESO, obtendo o resultado final em apenas 29 passos, sendo que este sozinho levaria 86 passos para completar a otimização a partir do início, um ganho de quase 2/3 de trabalho computacional. As estruturas finais obtidas pelos dois métodos foram idênticas. Tal resultado foi obtido de maneira prévia para apenas um exemplo, pode ser melhor explorado e validado.

6. CONCLUSÃO

As técnicas de *deep learning*, assim como o desenvolvimento plataformas e arquiteturas neurais específicas permitiram o rápido avanço da aplicabilidade de redes neurais artificiais em diversos problemas e áreas de pesquisa, inclusive em física ciência de materiais.

A aplicação de redes neurais convolucionais de arquitetura U-Net ao problema de otimização topológica estrutural tem mostrado resultados promissores, como demonstrado em trabalhos anteriores no campo e nos resultados deste, reduzindo o custo computacional dos métodos iterativos com baixa perda de performance das estruturas previstas.

Demonstramos que o uso de sensibilidade condicionada como entrada da rede é mais direto e produz resultados similares ao uso de topologia com gradiente. Também percebemos que este performa muito melhor que entrada de topologia em situações de baixa otimização.

Visualizamos um futuro na área em que os métodos de otimização topológica não serão limitados a métodos iterativos ou a redes neurais artificiais, e sim à combinação das técnicas, empregando o melhor de cada em um único algoritmo integrado.

7. BIBLIOGRAFIA

[1] - Sosnovik, Ivan, Oseledets, Ivan; Russian Journal of Numerical Analysis and Mathematical Modelling, 34(4); 215(2019);

[2] - Omonte Neves, João Vítor; "*Utilização de Métodos de Inteligência Artificial em Problemas de Otimização Topológica*"; UNICAMP; Campinas, SP, Brasil; 2020;

[3] - Fromlabs; *"Topology Optimization 101: How to Use Algorithmic Models to Create Lightweight Design"*; https://formlabs.com/blog/topology-optimization/; 2022;

[4] – Helm, Mark; *"Wild Cucumber"*; https://www.photomacrography.net/forum/viewtopic.php?t=11111; 2022;

[5] – Kwon, Young W., Bang, Hyochoong; "*The finite element method using MATLAB*"; CRC Press; Boca Raton, Florida, Estados Unidos da América; 1996;

[6] - Bendsøe, M.P., Kikuchi, N; Computer Methods in Applied Mechanics and Engineering 71(2);197(1988);

[7] – Xie, Y.M., Huang, X.; "*Evolutionary Topology Optimization of Continuum Structures: Methods and Applications*"; John Wiley & Sons; Chichester, West Sussex, Reino Unido; 2010;

[8] – Wikipedia; *"Photoelasticity"*; https://en.wikipedia.org/wiki/Photoelasticity; 2022;

[9] – Aggarwal, Charu C.; "*Neural Networks and Deep Learning: a Textbook*"; Springer International Publishing AG, Suíça; 2018;

[10] – Goodfellow, Ian, Bengio, Yoshua, Courville, Aaron; "*Deep Learning*"; MIT Press; Cambridge, MA, Estados Unidos da América; 2016;

[11] – Godoy, Daniel; *"Understanding binary cross-entropy / log loss: a visual explanation"*; https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a; 2022;

[12] - Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, Salakhutdinov, Ruslan; *"Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research"*; 15; 1929-1958 (2014);

[13] - Yen-Chi Chen, Tzu-Chieh Wei, Zhang, Yu, Yoo; *"Quantum Convolutional Neural Networks for High Energy Physics Data Analysis";* CoRR, abs/2012.12177; 2020;

[14] - Shafkat, Irhum; "*Intuitively Understanding Convolutions for Deep Learning, Towards Data Science";* https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1; 2022;

[15] – Purohit, Rahul; "*Fully Convolutional Network (Semantic Segmentation)"*; https://www.mygreatlearning.com/blog/fcn-fully-convolutional-network-semantic-segmentation/; 2022;

[16] - Ronneberger O., Fischer P., Brox T.; "*U-Net: Convolutional Networks for Biomedical Image Segmentation*"; Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg; Freiburg, Alemanha; 2015;

[17] - Oskal, K.R.J., Risdal, M., Janssen, E.A.M. et al; "*A U-net based approach to epidermal tissue segmentation in whole slide histopathological images".* SN Appl. Sci. 1, 672 (2019). https://doi.org/10.1007/s42452-019-0694-y;

[18] – Subramanyam, Vineeth S; *"IOU (Intersection over Union)"*; https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef; 2022;

[19] - Abadi, Mart; *"Tensorflow: A system for large-scale machine learning"*; 2015;

[20] - Kingma, Diederik P., Ba, Jimmy; *"Adam: A Method for Stochastic Optimization; International Conference for Learning Representations"*; San Diego, CA, Estados Unidos da América, 2015;

[21] – Bharath, K; *"Understanding ReLU: The Most Popular Activation Function in 5 Minutes!"*; https://towardsdatascience.com/understanding-relu-the-most-popular-activation-function-in-5-minutes-459e3a2124f; 2022;

[22] – Hunter, J. D.; *"Matplotlib: A 2D Graphics Environment"*; Computing in Science & Engineering; 9(3); 90-95 (2007);

[23] - Harris, C.R., Millman, K.J., van der Walt, S.J. et al; *"Array programming with NumPy"*. Nature 585; 357–362 (2020); DOI: 10.1038/s41586-020-2649-2;

[24] - McKinney, W., et al.; *"Data structures for statistical computing in python".* Proceedings of the 9th Python in Science Conference; 445; 51–56 (2010);