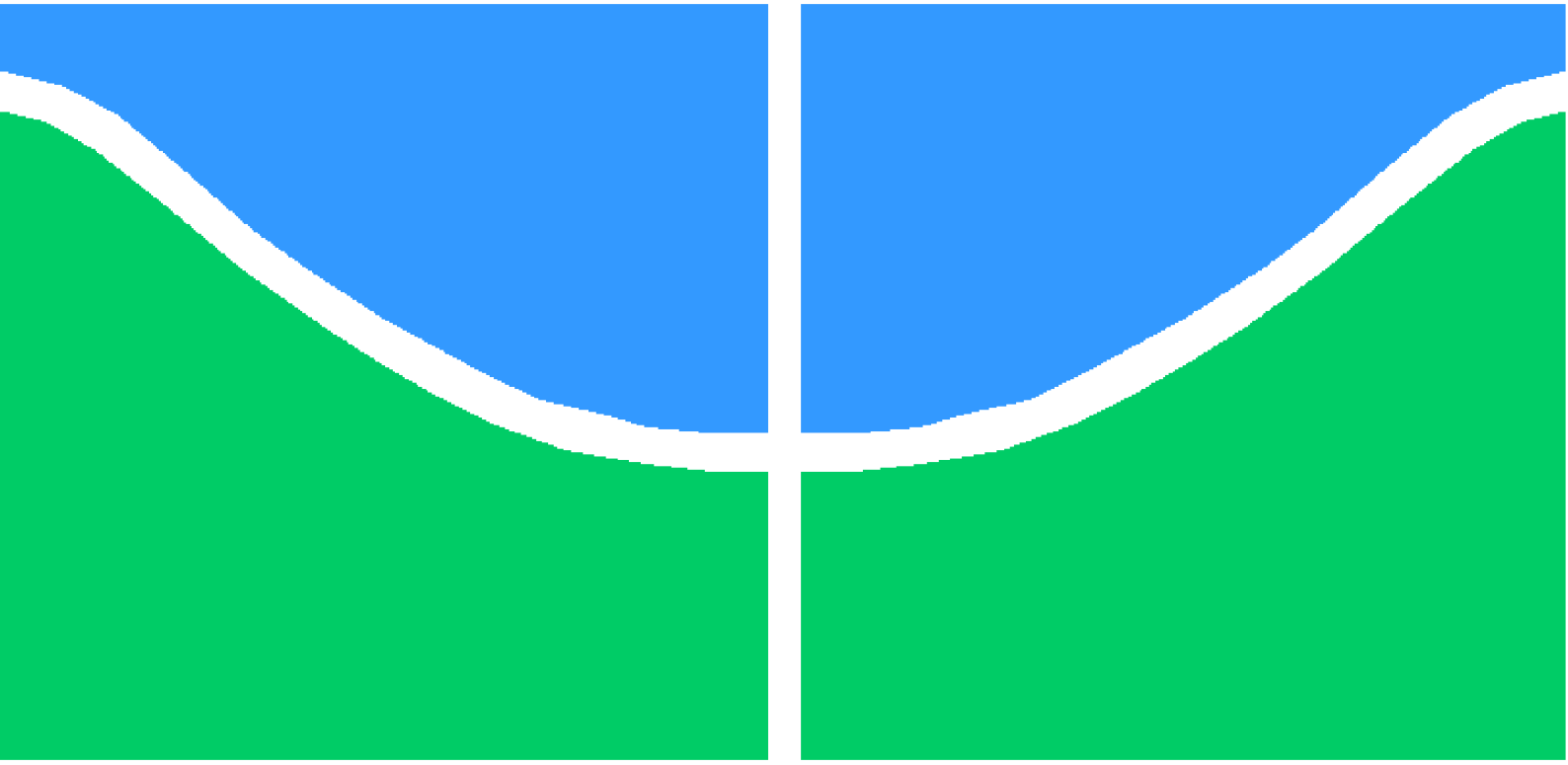


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia Eletrônica

Antena Holográfica Reconfigurável Com Otimização de Apontamento de Feixe Pelo Uso de Redes Neurais e Otimização Convexa

Autor: Wemerson Fontenele Sousa

Orientador: Dr. Sébastien Roland Marie Joseph Rondineau

Brasília, DF

2021

Wemerson Fontenele Sousa

Antena Holográfica Reconfigurável Com Otimização de Apontamento de Feixe Pelo Uso de Redes Neurais e Otimização Convexa

Monografia submetida ao curso de graduação em Engenharia Eletrônicada Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Sébastien Roland Marie Joseph Rondineau

Coorientador: Dr. Daniel Costa Araújo

Brasília, DF

2021

Wemerson Fontenele Sousa

Antena Holográfica Reconfigurável Com Otimização de Apontamento de Feixe Pelo Uso de Redes Neurais e Otimização Convexa/ Wemerson Fontenele Sousa. – Brasília, DF, 2021-

110 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Sébastien Roland Marie Joseph Rondineau

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2021.

1. Holografia. 2. Redes Neurais. I. Dr. Sébastien Roland Marie Joseph Rondineau. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Antena Holográfica Reconfigurável Com Otimização de Apontamento de Feixe Pelo Uso de Redes Neurais e Otimização Convexa

CDU 02:141:005.6

Wemerson Fontenele Sousa

Antena Holográfica Reconfigurável Com Otimização de Apontamento de Feixe Pelo Uso de Redes Neurais e Otimização Convexa

Monografia submetida ao curso de graduação em Engenharia Eletrônicada Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 08 de novembro de 2021:

Dr. Sébastien Roland Marie Joseph Rondineau
Orientador

Prof. Dr. Daniel Costa Araújo
Convidado 1

Prof. Dr. Leonardo Aguayo
Convidado 2

Brasília, DF
2021

Este trabalho é dedicado a minha família, que sempre me apoiou ao longo desta difícil jornada da graduação.

Agradecimentos

Gostaria de agradecer primeiramente à minha família que independente da situação, sempre me apoiou e me deu suporte para que eu pudesse chegar a esta etapa de conclusão de curso.

Agradeço também ao orientador Sébastien, que deu suporte para a conclusão deste trabalho, além de incentivos prévios para que eu pudesse direcionar meu curso à área de telecomunicações. E ao co-orientador Daniel, que me auxiliou com ensinamentos e orientações para o desenvolvimento deste trabalho.

Além disso, gostaria de agradecer aos meus amigos e colegas que fiz ao longo do curso, e a todos os professores com quem tive o prazer de estudar.

Resumo

Este trabalho retrata o desenvolvimento de uma antena reconfigurável em banda Ku, com frequência central em 12 GHz, que se utiliza de técnicas de holografia e metasuperfície para o apontamento de feixe. Ademais, conceitos de rede neural convolucional e otimização convexa são aplicados de modo a melhorar o apontamento de feixe. A antena apresentada é composta por dipolos do tipo fenda como elementos irradiantes, onde a cada elemento irradiante são adicionados diodos PIN, que são utilizados para controlar quais elementos devem ou não irradiar, gerando assim, sua reconfigurabilidade. Os elementos irradiantes são pequenos em termos do comprimento de onda, para que, com o uso de técnicas de metasuperfície, seja possível obter os apontamentos desejados. Para a definição da configuração da antena para cada feixe desejado, é utilizado o método da retro-propagação juntamente com técnicas de holografia. Com o intuito de se obter melhores resultados, foi utilizada uma rede neural convolucional, aplicada ao método da retro-propagação, e foram aplicadas técnicas de otimização convexa ao problema, de modo a obter um feixe ótimo. Foram utilizadas duas técnicas de otimização para que fosse possível analisá-las e compará-las, de modo a identificar as vantagens de cada uma. Tanto a antena, quanto a rede neural convolucional e o algoritmo de otimização convexa foram desenvolvidos e simulados de modo a obter resultados satisfatórios. Entretanto, a antena ainda não está em seu estado definitivo, uma vez que são necessárias alterações para que esta possa ser fabricada.

Palavras-chaves: Holografia. Metasuperfícies. Antena reconfigurável. Redes neurais convolucionais. Otimização convexa.

Abstract

This work describes the development of a reconfigurable Ku band antenna, with central frequency at 12 GHz, that uses holography and metasurface techniques for beamforming. Furthermore, convolutional neural network and convex optimization concepts are applied in order to improve the beamforming. The presented antenna is made up of slot dipoles as radiating elements, where at each radiating element is placed a PIN diode, which is used to control which elements should or should not radiate, thus generating the antenna reconfigurability. The radiating elements are small in terms of the wavelength, so that, with the use of meta-surface techniques, it is possible to obtain the desired beam. To define the antenna configuration for each desired beam, the back-propagation method is used alongside holography techniques. In order to obtain better results, a convolutional neural network was applied to the back-propagation method, and there were applied convex optimization techniques to the problem, in order to obtain an optimal beam. Two optimization techniques were used so that it was possible to analyze and compare them, in order to identify the advantages of each one. Both the antenna, the convolutional neural network and the convex optimization algorithm were developed and simulated in order to obtain satisfactory results. Nonetheless, the antenna is not yet in its final state, as changes are needed for it to be able to be manufactured.

Key-words: Holography. Metasurfaces. Reconfigurable antenna. Convolutional neural network. Convex optimization.

Lista de ilustrações

Figura 1 – Demarcação dos lóbulos de radiação em um padrão de radiação 22
Figura 2 – Exemplo de uma antena dipolo do tipo fenda 25
Figura 3 – Estrutura geral de um diodo PIN 28
Figura 4 – Circuito equivalente para o diodo PIN reversamente e diretamente polarizado (a) Estado reversamente polarizado. (b) Estado diretamente polarizado. 28
Figura 5 – Passos do processo de holografia ótica 30
Figura 6 – Processo de emissão de feixe por uma metasuperfície 33
Figura 7 – Exemplo de aplicação de conectividade esparsa, quando aplicada uma convolução com um núcleo de largura três 36
Figura 8 – Exemplo de aplicação do processo de *max pooling* com dimensão 2x2 a uma matriz 4x4 37
Figura 9 – Conector de borda e guia de ondas coplanares utilizados para alimentar a antena 42
Figura 10 – Funcionamento do componente chaveador sobre o elemento irradiante . 43
Figura 11 – Antena dipolo do tipo fenda desenvolvida 44
Figura 12 – Vista superior da antena 46
Figura 13 – Vista inferior da antena 46
Figura 14 – Vista isométrica da antena 46
Figura 15 – Vista superior da antena com 20x20 elementos 47
Figura 16 – Fase de H_x em cada elemento da metasuperfície, da onda de referência, em graus, para a antena com 11x11 elementos. 49
Figura 17 – Fase de H_x em cada elemento da metasuperfície, da onda de referência, em graus, para a antena com 20x20 elementos. 50
Figura 18 – Sumário da CNN desenvolvida 53
Figura 19 – Esquemático da CNN desenvolvida 53
Figura 20 – Cálculo do holograma para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 0$ e $\phi = 0$. (a) Fase obtida através da retropropagação em cada elemento da metasuperfície em graus. (b) Holograma encontrado do padrão de interferência onde em 1 (branco) são as antenas que devem irradiar e em 0 (preto) são as antenas que não devem irradiar. 56
Figura 21 – Diagramas de radiação simulados no HFSS para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 0$ e $\phi = 0$. (a) Diretividade em dB. (b) Diretividade em escala linear. 56

Figura 22 – Cálculo do holograma para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 45$ e $\phi = 90$. (a) Fase obtida através da retro-propagação em cada elemento da metasuperfície em graus. (b) Holograma encontrado do padrão de interferência onde em 1 (branco) são as antenas que devem irradiar e em 0 (preto) são as antenas que não devem irradiar. 57
Figura 23 – Diagramas de radiação simulados no HFSS para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 45$ e $\phi = 90$. (a) Diretividade em dB. (b) Diretividade em escala linear. 57
Figura 24 – Cálculo do holograma para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 45$ e $\phi = 45$. (a) Fase obtida através da retro-propagação em cada elemento da metasuperfície em graus. (b) Holograma encontrado do padrão de interferência onde em 1 (branco) são as antenas que devem irradiar e em 0 (preto) são as antenas que não devem irradiar. 58
Figura 25 – Diagramas de radiação simulados no HFSS para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 45$ e $\phi = 45$. (a) Diretividade em dB. (b) Diretividade em escala linear. 58
Figura 26 – Cálculo do holograma para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 30$ e $\phi = 120$. (a) Fase obtida através da retro-propagação em cada elemento da metasuperfície em graus. (b) Holograma encontrado do padrão de interferência onde em 1 (branco) são as antenas que devem irradiar e em 0 (preto) são as antenas que não devem irradiar. 59
Figura 27 – Diagramas de radiação simulados no HFSS para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 30$ e $\phi = 120$. (a) Diretividade em dB. (b) Diretividade em escala linear. 59
Figura 28 – Fator de Arranjo Obtido Pelo Método da Retro-Propagação. (a) Calculado. (b) Com Lóbulos Secundários Reduzidos Manualmente. 61
Figura 29 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 23$ e $\phi = 230$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. 62
Figura 30 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 23$ e $\phi = 230$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. 63
Figura 31 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 10$ e $\phi = 115$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. 63

Figura 32 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 10$ e $\phi = 115$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. 63
Figura 33 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 25$ e $\phi = 180$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. 64
Figura 34 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 25$ e $\phi = 180$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. 64
Figura 35 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 50$ e $\phi = 0$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. (c) Obtido pelo método da otimização convexa. 65
Figura 36 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 50$ e $\phi = 0$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN (c) Obtido pelo método da otimização convexa. . 65
Figura 37 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 0$ e $\phi = 0$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. (c) Obtido pelo método da otimização convexa. 66
Figura 38 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 0$ e $\phi = 0$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN (c) Obtido pelo método da otimização convexa. . 66
Figura 39 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 15$ e $\phi = 240$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. (c) Obtido pelo método da otimização convexa. 66
Figura 40 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 15$ e $\phi = 240$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN (c) Obtido pelo método da otimização convexa. 67

Lista de tabelas

Tabela 1 – Ficha de dados do laminado Kappa438 41
Tabela 2 – Comparação entre os possíveis componentes de controle para a antena selecionada 45
Tabela 3 – Dimensões calculadas para as variáveis de comprimento da antena . . . 47

Lista de abreviaturas e siglas

3D	Tridimensional
2D	Bidimensional
CC	Corrente Contínua
CNN	Rede neural convolucional
CPW	Guia de ondas coplanares
FET	Transistor de efeito de campo
FNBW	Primeira largura de feixe nula
HFSS	Simulador estrutural de alta frequência
HPBW	Largura de feixe de metade da potência
IL	Perda de inserção
MIMO	Múltiplas entradas múltiplas saídas
PEC	Condutor elétrico perfeito
RF	Rádiofrequência
RF-MEMS	Sistema microeletromecânico de radiofrequência

Lista de símbolos

$AF(\theta, \phi)$	Fator de arranjo
c	Velocidade da luz
C	Capacitância
$D(\theta, \phi)$	Diretividade da antena
f	Frequência
F	Farad
$G(\theta, \phi)$	Ganho da antena
H_x	Componente X do campo magnético
i	Linha de uma matriz
I	Imagem bidimensional
$I(x, y)$	Intensidade do padrão de interferência do holograma
j	Coluna de uma matriz
k	Número de onda
K	Núcleo de convolução
L	Indutância
M	Número de linhas de uma matriz
N	Número de colunas de uma matriz
O_0	Amplitude da onda de objeto
P_{rad}	Potência total irradiada
R_0	Amplitude da onda de referência
$S(i, j)$	Resultado de uma convolução
$S(\theta, \phi)$	Densidade de potência radiada
T_t	Tempo de exposição do holograma à onda de referência

$T(x, y)$	Amplitude da onda transmitida pelo holograma
$U(x, y)$	Onda emitida pelo holograma durante reconstrução da imagem
β_t	Constante de proporcionalidade
ϵ_L	Eficiência da antena
ϵ_r	Permissividade elétrica do material
Γ	Coeficiente de reflexão
λ	Comprimento de onda
λ_g	Comprimento de onda guiada
ω	Frequência angular
Ω	Ohm
ϕ	Ângulo em coordenadas esféricas
ϕ_o	Fase da onda de objeto
θ_r	Angulação entre onda de referência e eixo z
θ	Ângulo em coordenadas esféricas

Sumário

1 INTRODUÇÃO 18

Introdução 18
1.1 Contextualização e Problematização 18
1.2 Justificativa 19
1.3 Objetivos 19
1.3.1 Objetivo Geral 19
1.3.2 Objetivos Específicos 20
1.4 Estrutura da Dissertação 20

2 FUNDAMENTAÇÃO TEÓRICA 21
2.1 Antenas 21
2.1.1 Parâmetros de Antenas 21
2.1.1.1 Padrão de Radiação 21
2.1.1.2 Diretividade e Ganho 22
2.1.2 Arranjo de Antenas 23
2.1.3 Antenas Reconfiguráveis 24
2.1.3.1 Técnicas de Reconfiguração de Feixe 24
2.1.4 Antenas Dipolo do Tipo Fenda 25
2.2 Diodos PIN 27
2.2.1 Funcionamento 27
2.3 Holografia 29
2.3.1 Princípio da Holografia 29
2.3.2 Descrição Matemática do Fenômeno 30
2.4 Metasuperfícies 32
2.5 Redes Neurais Convolucionais 34
2.5.1 Operações Realizadas por uma CNN 34
2.5.1.1 Convolução 35
2.5.1.2 ReLU 36
2.5.1.3 Pooling 37
2.5.2 Camada Totalmente Conectada 37
2.5.3 Softmax 38
2.6 Otimização Convexa 39
2.6.1 Otimização Convexa Sobre o Fator de Arranjo 39

3 DESENVOLVIMENTO DA ANTENA 41

3.1 Antena de Metasuperfície . . . 41
3.2 Meio Propagante . . . 41
3.3 Alimentador . . . 42
3.4 Elemento Irradiante . . . 42
3.5 Componente de Controle . . . 44
3.6 Integração da Antena . . . 45
3.6.1 Dimensões da Antena . . . 47
3.7 Procedimento Holográfico . . . 47
3.7.1 Cálculos Realizados . . . 48
3.7.2 Implementação do Procedimento Holográfico . . . 49

4 DESENVOLVIMENTO DA REDE NEURAL CONVOLUCIONAL . . . 51
4.1 Entradas e Saídas da Rede Neural Convolucional . . . 51
4.2 Estrutura da Rede Neural Convolucional . . . 52

5 RESULTADOS E ANÁLISES . . . 55
5.1 Apontamento de Feixe pelo Método da Retro-Propagação . . . 55
5.1.1 Apontamento de feixe,em coordenadas esféricas, para os ângulos . . . 55
5.1.2 Apontamento de feixe,em coordenadas esféricas, para os ângulos . . . 56
5.1.3 Apontamento de feixe,em coordenadas esféricas, para os ângulos . . . 57
5.1.4 Apontamento de feixe,em coordenadas esféricas, para os ângulos . . . 58
5.1.5 Análise dos Resultados . . . 59
5.2 Processo de Otimização . . . 60
5.2.1 Resultados Para a Antena 11x11 . . . 62
5.2.2 Resultados Para a Antena 20x20 . . . 64

6 CONCLUSÃO . . . 68
6.1 Trabalhos Futuros . . . 69

REFERÊNCIAS . . . 70

APÊNDICES 72

APÊNDICE A – CÓDIGOS DESENVOLVIDOS PARA O CÁLCULO DO HOLOGRAMA DA ANTENA PELO MÉTODO DA RETRO PROPAGAÇÃO . . . 73

APÊNDICE B – CÓDIGOS DESENVOLVIDOS PARA A CONSTRUÇÃO DA BASE DE DADOS PARA TREINO E TESTES DA CNN . . . 85

APÊNDICE C – CÓDIGOS DESENVOLVIDOS PARA A CNN . . . 92

1 Introdução

1.1 Contextualização e Problematização

Antenas reconfiguráveis podem orientar o apontamento do feixe radiado de maneira controlada, e reversível, sem a necessidade do uso de movimentações mecânicas (YURDUSEVEN et al., 2017). O fato de não serem realizadas movimentações mecânicas faz com que estas antenas possam ter uma maior durabilidade e maior velocidade de reconfiguração (na casa dos nanosegundos) que as antenas movidas mecanicamente (na casa dos segundos). Antenas reconfiguráveis possuem, também, grande variedade de aplicações, que vão desde comunicações via satélite, até dispositivos biomédicos (MOHANTA; KOUZANI; MANDAL, 2010).

Em anos recentes, metasuperfícies vem recebendo grande reconhecimento e sendo utilizadas para um grande alcance de aplicações, estas podem ser exploradas em diversas regiões do espectro eletromagnético, mas com principais aplicações nas frequências de micro-ondas, como pode ser visto em FAENZI et al. (2019), OVEJERO e MACI (2015), FONG et al. (2010). Seu uso se deve ao fato de metasuperfícies conseguirem apresentar características não encontradas nos métodos tradicionais de desenvolvimento de circuitos de micro-ondas (FAENZI et al., 2019).

Técnicas de holografia são aplicadas no desenvolvimento destas metasuperfícies, como apresentado por FONG et al. (2010), onde estes se utilizam da onda conhecida utilizada para alimentação da antena, e ao encontrarem o padrão de interferência entre esta e os campos associados com a radiação desejada, define-se o o padrão de superfície necessário para que seja possível a geração desta radiação, este então cria uma metasuperfície de elementos passivos de modo a atingir esse padrão desejado. Técnica semelhante ao utilizado por YURDUSEVEN et al. (2017), com a diferença que este utiliza elementos ativos para modificar o padrão da superfície, de modo a conseguir gerar diversos feixes, ao invés de um único feixe pré-definido.

Redes neurais convolucionais vem obtendo grande sucesso no reconhecimento de padrões em imagens nos últimos anos (SIMONYAN; ZISSERMAN, 2015). Tal fator foi utilizado e apresentado por TAO et al. (2020), onde é aproveitado o fato da retro-propagação do feixe sobre a antena reconfigurável desenvolvida apresentar padrões bem definidos, para que fosse possível sua aplicação em uma rede neural convolucional, sendo que nesta são aplicados para seu treinamento, os dados da retro propagação dos feixes formados e a configuração da antena necessária para tal formação, com isso, é possível passar como entradas a esta rede neural convolucional os dados dos feixes desejados, de modo a conseguir

obter em sua saída, a configuração necessária a ser aplicada à antena para a formação deste feixe.

Otimização convexa é um um campo da matemática bem estabelecido e com aplicações em diversas áreas. Para o caso de otimização de apontamento de feixe de antenas reconfiguráveis, essa abordagem também já possui seu funcionamento consolidado, com aplicações em diferentes tipos de antenas, como, por exemplo, o trabalho desenvolvido por FUCHS e RONDINEAU (2016), que a aplica para otimizar o controle das excitações de um arranjo de antenas, de modo a obter um feixe dentro das margens desejadas. Tal abordagem, também, foi aplicada por SANTANA (2021), que utiliza a otimização convexa para definir a combinação de antenas que devem irradiar potência, de modo a se obter um feixe específico otimizado, dentro dos critérios estabelecidos.

1.2 Justificativa

Antenas reconfiguráveis eletronicamente possuem vantagens em relação a antenas tradicionais, como o fato de dispensarem a necessidade da realização de movimentações mecânicas, o que faz com que estas antenas possam ter uma maior durabilidade e maior velocidade de reconfiguração que as antenas movidas mecanicamente. E, apesar de existirem diferentes antenas reconfiguráveis eletronicamente, como apresentado em (MOHANTA; KOUZANI; MANDAL, 2010), esta foi escolhida devido ao uso de diodos PIN juntamente com técnicas de matasuperfície e holografia, que permitirem o desenvolvimento de uma antena com uma menor perda de inserção em altas frequências se comparadas com aquelas que usam transistores de efeito de campo (FET), e uma maior velocidade de chaveamento em relação a sistemas microeletrônicos de radiofrequência (RF-MEMS) (KEYSIGHT, 2017). E, se comparado com outras técnicas de reconfiguração de feixe, esta possui um custo menor que aplicações como arranjo de fases e o método de múltiplas entradas e múltiplas saídas (MIMO) (BLACK, 2017).

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver uma antena reconfigurável, que se utiliza de técnicas de metasuperfícies e holografia para o apontamento de feixe, além da criação uma rede neural convolucional e sua aplicação ao método de apontamento de feixe e da aplicação da técnica de otimização convexa ao problema, para que fosse possível obter uma otimização na configuração da antena, e comparar ambas técnicas de otimização de modo a obter as vantagens de cada uma.

1.3.2 Objetivos Específicos

1. Apresentar a fundamentação teórica por trás do desenvolvimento deste trabalho;
2. Desenvolver uma antena reconfigurável que permita a alteração de seu diagrama de radiação através da modificação da polarização de diodos PIN;
3. Desenvolver uma rede neural convolucional para ser aplicada ao método de apontamento de feixe, de modo a otimizá-lo;
4. Desenvolver um algoritmo de otimização convexa para o problema de apontamento de feixe;
5. Realizar simulações do trabalho realizado, de modo a avaliar seu funcionamento;
6. Comparar as duas técnicas de otimização de feixe aplicadas de modo a identificar as vantagens de cada uma.

1.4 Estrutura da Dissertação

O trabalho está organizado de maneira a acompanhar os passos do desenvolvimento do projeto. No Capítulo 2, está apresentada a fundamentação teórica necessária para a compreensão do projeto, contendo explicações acerca de teoria de antenas, de metasuperfícies, de holografia, e a teoria sobre redes neurais convolucionais. No Capítulo 3 está apresentado como foi desenvolvido o circuito referente à antena. No Capítulo 4 está apresentado o desenvolvimento da rede neural convolucional. No Capítulo 5 estão apresentados os resultados do projeto, assim como uma breve análise deste. No Capítulo 6 está apresentado um resumo do que foi desenvolvido durante esta etapa do projeto, assim como comentários sobre os próximos passos a serem tomados para sua finalização.

2 Fundamentação Teórica

2.1 Antenas

Antenas são definidas como sendo a estrutura de transição entre as ondas eletromagnéticas no espaço livre e as linhas de transmissão, funcionando tanto como elemento irradiante, quanto como elemento receptor destas ondas, ao converter fótons em elétrons ou elétrons em fótons (BALANIS, 2005).

2.1.1 Parâmetros de Antenas

O desempenho de uma antena pode ser explicado através de diversos parâmetros, e dentre estes parâmetros, alguns são relacionados e outros não. Para este trabalho, os principais parâmetros levados em consideração são o padrão de radiação, a diretividade e o ganho.

2.1.1.1 Padrão de Radiação

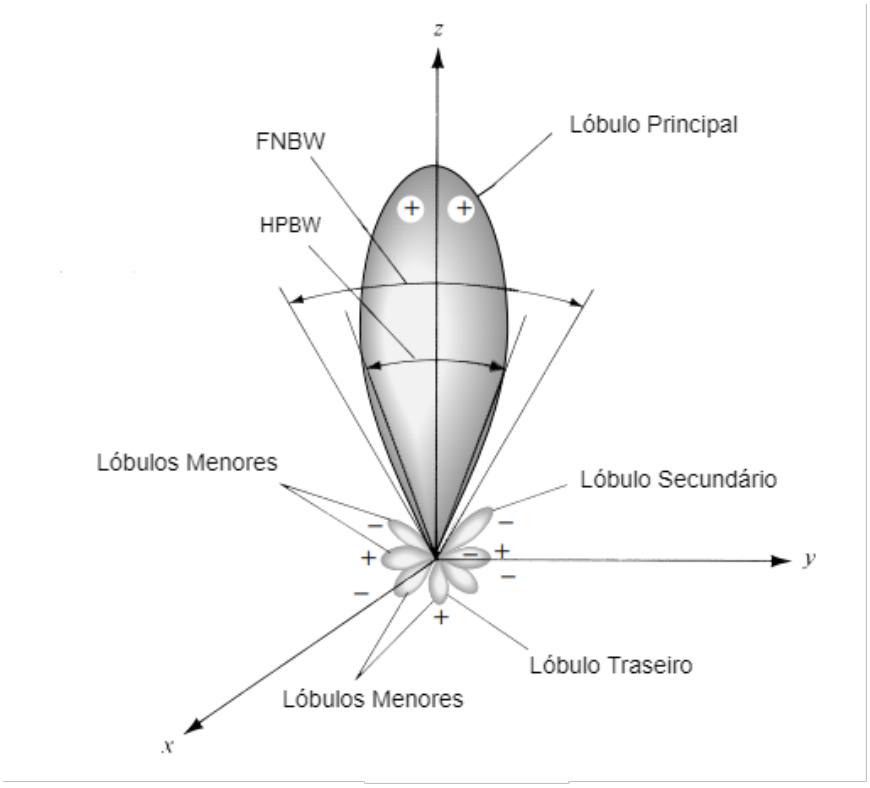
O padrão de radiação de uma antena é definido como a função matemática ou representação gráfica das propriedades de radiação da antena como uma função de coordenadas espaciais. O padrão de radiação geralmente é determinado em campo distante como função de coordenadas direcionais e após sua determinação é possível obter características como diretividade, intensidade de radiação e potência radiada (BALANIS, 2005).

As várias partes do diagrama de radiação são chamados de lóbulos, que se tratam de uma porção do campo radiado, onde suas bordas possuem baixa intensidade de radiação e podem ser classificados em lóbulo principal, lóbulo secundário, lóbulos menores e lóbulo traseiro (BALANIS, 2005). O diagrama de radiação apresentado na Figura 1, apresenta como estes lóbulos são identificados em um padrão de radiação. O lóbulo principal é o lóbulo de maior intensidade de radiação, e em algumas antenas podem ocorrer mais de um lóbulo principal. O lóbulo secundário é um lóbulo que pode apontar em qualquer direção diferente a do lóbulo principal. Um lóbulo traseiro é aquele cujo eixo faz um ângulo de 180º com o ângulo principal. Lóbulos menores representam radiação em direções indesejadas, e devem ser minimizados.

A Figura 1 apresenta ainda a marcação de duas faixas no diagrama, que são a largura de feixe de metade da potência (HPBW de *half power beamwidth*) que indica o ângulo entre as duas direções onde a intensidade do feixe é igual a metade da máxima intensidade do feixe, que em um diagrama em decibéis (dB), indicaria uma queda de 3

dB, a outra marcação é a primeira largura para a qual o feixe é nulo (FNBW de *first null beamwidth*), que indica a a separação entre as duas regiões do padrão de radiação onde esta é considerada mínima, tendo em seu centro o feixe de maior intensidade.

Figura 1 – Demarcação dos lóbulos de radiação em um padrão de radiação



Fonte: BALANIS (2005)

2.1.1.2 Diretividade e Ganho

Uma importante característica de qualquer antena é sua capacidade de concentrar energia em uma direção, reduzindo a radiação em outras regiões, esta característica é a chamada diretividade, e para uma antena 100% eficiente, esta quantia é igual ao ganho da antena (STUTZMAN; THIELE, 2013).

Para entender a matemática por trás da diretividade de uma antena, é necessário primeiro entender como é calculada a radiação emitida por esta. A densidade de potência radiada para uma antena omnidirecional, é dada pela Equação (2.1) onde R é a distância entre a antena e o ponto de observação e P_{rad} é a potência total radiada por esta (MAILLOUX, 2018).

$$S(\theta, \phi) = \frac{P_{rad}}{4\pi R^2} \tag{2.1}$$

Para uma antena direcional, entretanto, a densidade de potência radiada varia de acordo com a direção, então, considerando um sistema de coordenadas esféricas, essa

densidade é tratada como apresentada na Equação (2.2), onde esta varia a depender de sua diretividade.

$$S(\theta, \phi) = \frac{P_{rad} D(\theta, \phi)}{4\pi R^2} \tag{2.2}$$

Com isso, é possível encontrar a diretividade de uma antena como dada pela Equação (2.3).

$$D(\theta, \phi) = \frac{4\pi R^2 S(\theta, \phi)}{P_{rad}} \tag{2.3}$$

Que pode ser reescrita como apresentado na Equação (2.4 (MAILLOUX, 2018)).

$$D(\theta, \phi) = \frac{4\pi S(\theta, \phi)}{\int_0^{2\pi} \int_0^{\pi} S(\theta, \phi)\, d\phi\, d\theta} \tag{2.4}$$

Já o ganho, é relacionado a diretividade, e pode ser encontrado pela Equação 2.5, onde ϵ_L é um fator de eficiência que considera as perdas do circuito, e Γ é o coeficiente de reflexão da antena, medido na linha de transmissão.

$$G(\theta, \phi) = \epsilon_L (1 - |\Gamma|^2) D(\theta, \phi) \tag{2.5}$$

2.1.2 Arranjo de Antenas

Em muitas situações são necessárias determinadas características das antenas, onde um único elemento não é o suficiente. Entretanto, em algumas situações, é possível atingir essas características com o uso de um conjunto de elementos irradiantes. Este conjunto é chamado de arranjo de antenas, e pode ser tratado como uma única antena (BALANIS, 2005).

Para o arranjo de antenas, é possível identificar seu padrão de radiação com o uso da Equação (2.6), cujo cálculo necessita de dois dados: $AF(\theta, \phi)$, o fator de arranjo do conjunto de elementos e $\vec{e}(\theta, \phi)$, o padrão de radiação dos elementos irradiantes (BHATTACHARYYA, 2006).

$$\vec{E}(\theta, \phi) = \vec{e}(\theta, \phi) AF(\theta, \phi) \tag{2.6}$$

O fator de arranjo, para uma antena com MxN elementos pode ser obtido a partir da Equação (2.7), na qual, $k = 2\pi/\lambda$ é o número de onda, A_{mn} é a matriz de excitação das antenas e (x_{mn}, y_{mn}) representa as coordenadas dos elementos irradiantes do arranjo (BHATTACHARYYA, 2006).

$$AF(\theta,\phi) = \sum_{m=1}^{M}\left(\sum_{n=1}^{N} A_{mn} e^{jk(x_{mn}sen(\theta)cos(\phi)+y_{mn}sen(\theta)sen(\phi))}\right) \tag{2.7}$$

Considerando uma antena com elementos irradiantes idênticos, estes teriam o mesmo padrão de radiação. Portanto, para modificar o padrão de radiação de um arranjo de antenas, é necessário modificar o seu fator de arranjo. E, como pode ser visto na Equação (2.7), o fator de arranjo pode ser alterado ao se modificar a matriz de excitação dos elementos irradiantes. Existem diversas maneiras de modificar a matriz de excitação dos elementos irradiantes, inclusive, de maneira reconfigurável eletronicamente (MOHANTA; KOUZANI; MANDAL, 2010).

2.1.3 Antenas Reconfiguráveis

Antenas reconfiguráveis eletronicamente podem orientar modificar sua polarização, seu ganho, número de feixes, direção de apontamento do feixe principal e diversos outros fatores de maneira controlada eletronicamente, sem a necessidade do uso de movimentações mecânicas, e por este fator, estas apresentam maior durabilidade, e maior velocidade de apontamento de feixe em determinada direção (na casa dos nanosegundos) que as antenas movidas mecanicamente (na casa dos segundos) (YURDUSEVEN et al., 2017). Tais antenas ganham destaque por tais fatores, e possuem grande variedade de aplicações, que vão desde comunicações via satélite, até dispositivos biomédicos (MOHANTA; KOUZANI; MANDAL, 2010).

2.1.3.1 Técnicas de Reconfiguração de Feixe

Dentre as principais técnicas de reconfiguração de feixe, estão a de multiplas entradas e multiplas saídas (MIMO, do inglês *multiple input multiple output*), formador de feixe holográfico, e o uso de arranjo de fases. Sendo que cada um destes possui vantagens em relação ao outro, portanto não é possível definir exatamente qual técnica é a melhor no geral, pois cada uma possui suas aplicabilidades (BLACK, 2017).

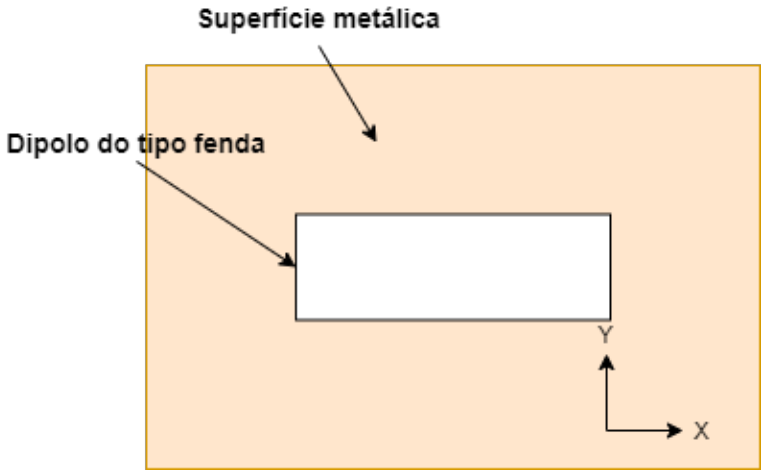
A tecnologia MIMO por exemplo apresenta o menor número de elementos irradiantes, porém utiliza componentes caros por trás de cada elemento, o que faz com que esta seja mais cara que as demais. Antenas reconfiguráveis que utilizam técnicas de holografia possuem o menor custo, tamanho, peso e potência dentre as técnicas de reconfiguração, porém, apresenta dificuldades em trabalhar com múltiplos feixes, se comparado as demais técnicas. Já antenas que usam arranjo de fases, possuem um preço intermediário, porém este preço pode aumentar bastante a depender do número de feixes a serem apontados ao mesmo tempo.

2.1.4 Antenas Dipolo do Tipo Fenda

Antenas do tipo dipolo são tradicionalmente uma estrutura em reta de um condutor, alimentada em seu centro, que são amplamente utilizadas por causa do seu padrão de corrente de ondas estacionárias (BALANIS, 2005). Antenas dipolo do tipo fenda, por outro lado, são uma espécie de antenas complementares ao dipolo tradicional, sendo formado por uma fenda em uma superfície metálica, e sua alimentação é dada de maneira diferente, através de uma onda guiada em sua superfície, ou abaixo de sua estrutura.

Um exemplo de uma antena dipolo do tipo fenda está apresentado na Figura 2, onde está apresentado uma superfície metálica em laranja, e em branco o dipolo do tipo fenda, indicando a ausência de metal. É importante ressaltar que devido a geometria do elemento ser uma espécie de retângulo, este irá acoplar apenas campos magnéticos no sentido ao qual seu lado de maior dimensão está orientado, no caso da figura, o eixo X. Além disso, devido ao elemento acoplar apenas campos magnéticos no sentido do eixo X, temos que o elemento será linearmente polarizado ao longo de X. Semelhante a isso, caso a orientação do elemento radiante estivesse rotacionada 90º (orientada no eixo Y), deveria ser usada apenas a componente Y, e caso o elemento estivesse em um meio termo entre essas orientações, ele acoplaria ambas as componentes X e Y (YURDUSEVEN et al., 2017).

Figura 2 – Exemplo de uma antena dipolo do tipo fenda



Fonte: Autoria Própria

Antenas dipolo do tipo fenda são antenas que possuem uma maior irradiação quando o comprimento de sua fenda possui comprimento $\lambda_g/2$, ou seja, metade do comprimento de onda guiado no qual esta deve irradiar, onde $\lambda = c/f$, onde f é a frequência de operação e c é a velocidade da luz, e $\lambda_g = \lambda/\sqrt{(\epsilon_r + 1)/2}$, onde ϵ_r é a constante dielétrica do material pelo qual a onda guiada se propaga. Apesar deste fator trazer um máximo acoplamento do modo guiado, o que aumenta a eficiência de radiação, isso introduz uma maior perturbação ao modo guiado criado para alimentar esta antena, portanto, para tra-

balhar com um arranjo de antenas, é recomendado que estes possuam um comprimento levemente menor (YURDUSEVEN et al., 2017).

2.2 Diodos PIN

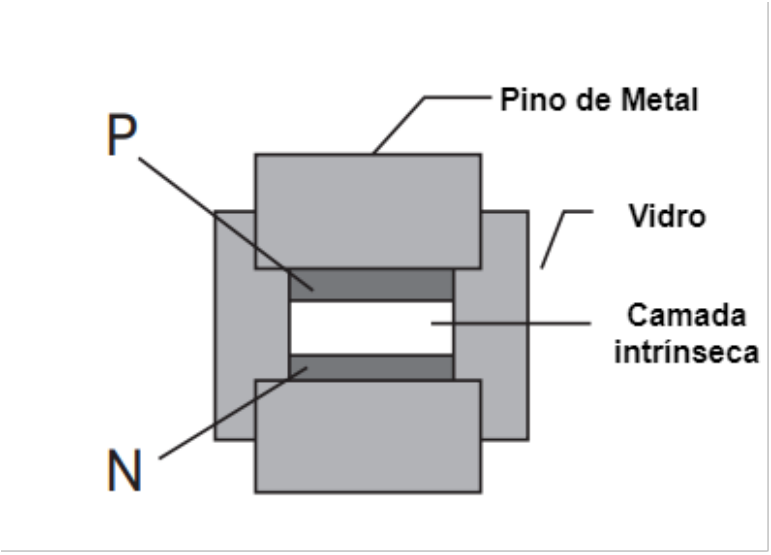
Chaveadores são utilizados extensivamente em sistemas de que trabalham nas frequências de micro ondas. Podendo ser utilizados para esta tarefa, chaveadores mecânicos, porém são volumosos e lentos. Diodos PIN, por outro lado, podem ser usados na construção de chaveadores eletrônicos planares, e possuem uma velocidade de chaveamento em alta velocidade, podendo ter uma taxa de chaveamento de até 20 ns (POZAR, 2012).

2.2.1 Funcionamento

Um diodo PIN é um dispositivo semicondutor controlado por corrente, funcionando como um resistor variável nas radiofrequências (RF) e frequências de micro ondas, que a depender da corrente que passa por ele, sua resistência pode variar de menos de 1 Ω, à 10 kΩ. Sua corrente de corrente contínua (CC) de controle é a única que interfere em sua resistência, e ao variar esta corrente que o alimenta entre um estado de ligado e desligado, o diodo PIN pode ser usado como chaveador (KEYSIGHT, 2017).

A estrutura de um diodo PIN pode ser vista na Figura 3. Nesta é possível ver que este é composto por duas camadas de semicondutores N e P, e contém uma camada intrínseca levemente dopada entre estes, além de uma camada de vidro envolvendo as demais camadas, que possui como principal função tornar o diodo PIN mais durável para aplicações de alta potência. Quando polarizado inversamente uma capacitância de junção leva este para um diodo de uma impedância relativamente alta, e quando este é diretamente polarizado, a capacitância é removida pela corrente de polarização direta, levando o diodo para um estado de baixa impedância. Essas características de um diodo PIN tornam este útil para aplicação em circuitos chaveadores (POZAR, 2012).

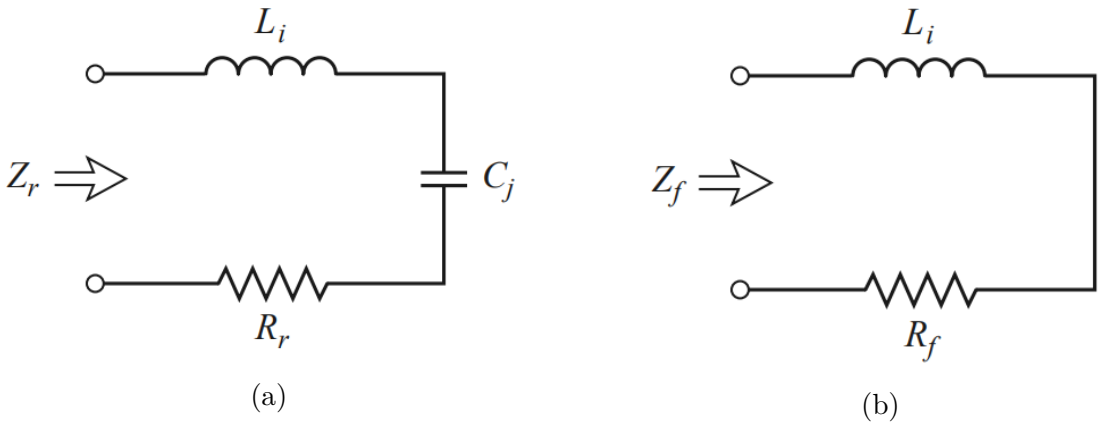
Figura 3 – Estrutura geral de um diodo PIN



Fonte: KEYSIGHT (2017)

Os circuitos equivalentes para os estados de polarização direta e reversa do diodo, estão apresentados na Figura 4. A indutância parasítica L_i, é geralmente menor que 1 nH, sua resistência reversa R_r é pequena em comparação com a resistência em série, e é geralmente ignorada devido à capacitância de junção. A corrente de polarização utilizada para mudar o estado do diodo está geralmente entre 10 mA e 30 mA. E sua tensão é de polarização reversa é tipicamente entre 10 e 60 V (POZAR, 2012).

Figura 4 – Circuito equivalente para o diodo PIN reversamente e diretamente polarizado (a) Estado reversamente polarizado. (b) Estado diretamente polarizado.



Fonte: POZAR (2012).

A tensão de polarização deve ser aplicada ao diodo juntamente com um circuito com estranguladores de sinais RF, e bloqueadores de sinais DC, para isolar a tensão de polarização CC do sinal RF (POZAR, 2012).

2.3 Holografia

A teoria da holografia, descoberta em 1948 por Dennis Gabor, é um procedimento através do qual é possível captar uma imagem tridimensional (3D) em um plano bidimensional (2D), de modo que ao se aplicar uma luz coerente a esse plano 2D, seja possível reconstruir a imagem captada (AMBS; HUIGNARD; LOISEAUX, 2005).

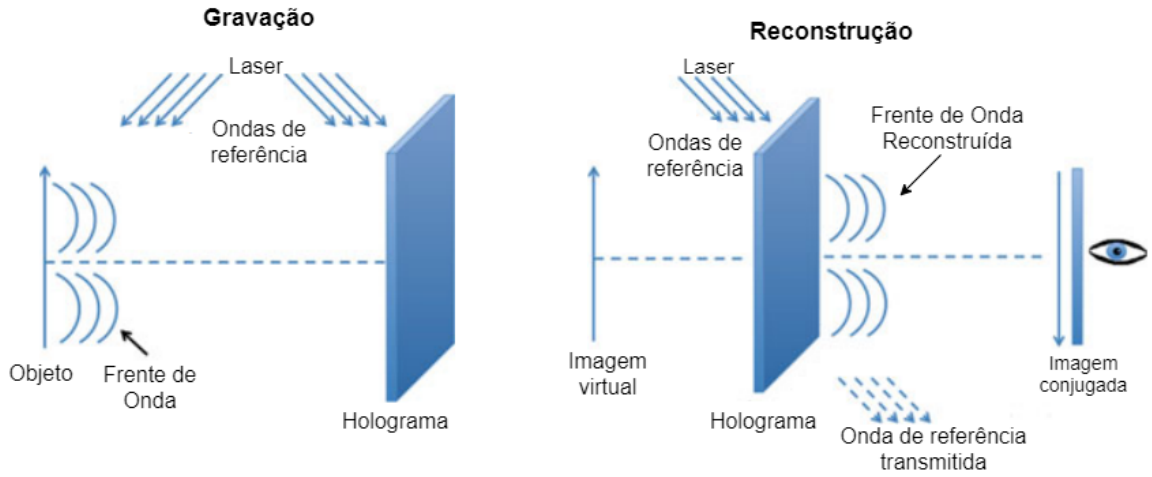
2.3.1 Princípio da Holografia

O propósito da holografia é a reconstrução de uma frente de onda. A qual é realizada através de um processo de dois passos, sendo eles a gravação da fase e da amplitude da frente de onda, e a sua reconstrução. Este processo é melhor explicado com o uso da Figura 5, onde são apresentados os passos de gravação e reconstrução no processo de holografia.

Para o processo de gravação, é aplicada uma onda de referência através de um laser de luz coerente (também chamada de onda de referência), que ilumina tanto o objeto, quanto a superfície de gravação do holograma. Para tal, geralmente é utilizado um separador para separar e difratar essa onda de referência para a superfície de gravação (LIZUKA, 2019). Após ser atingido pela luz de referência, o objeto gera uma frente de onda, a chamada onda de objeto, que também ilumina a superfície de gravação. A superfície de gravação ao ser iluminada pela onda de objeto e a onda de referência, grava o padrão de interferência, criando assim o holograma (RUSCH, 2016).

Já para o processo de reconstrução, o holograma que foi criado durante o processo de gravação deve ser iluminado pela mesma onda de referência utilizada durante sua gravação, de modo que esta onda de referência difrate ao iluminar o holograma, e algumas destas ondas geram uma cópia do objeto que havia sido gravado, também chamada de imagem virtual do objeto. Durante este processo, as ondas de referência se dividem em quatro conjuntos de feixes, o primeiro deles, chamado na imagem de onda de referência transmitida, é apenas a onda de referência que passa pelo holograma, o segundo deles acompanha a onda de referência e gera uma auréola ao seu redor, o terceiro deles é o conjunto de feixes que geram a imagem virtual, e o quarto conjunto de feixes gera uma cópia do objeto que havia sido gravado, porém, com fase inversa e, por isso, recebe o nome de imagem conjugada.

Figura 5 – Passos do processo de holografia ótica



Fonte: Autoria própria, adaptado de RUSCH (2016)

2.3.2 Descrição Matemática do Fenômeno

Para a descrição deste fenômeno, primeiro é necessário definir a onda de referência, que se trata de uma onda plana, onde considerando uma onda se propagando no plano (x,z) sua equação é definida em (2.8), onde θ_r é o ângulo entre a direção de propagação e o eixo z, $r_0(x)$ é a amplitude da onda, k é o número de onda (HARIHARAN, 2007).

$$R(x) = r_0(x)e^{jkxsen(\theta_r)} \tag{2.8}$$

É necessário, ainda, definir a onda de objeto. Porém, esta não possui uma forma única, visto que ela depende do formato do objeto. Então, para sua definição, a onda de objeto é considerada como uma onda genérica, como apresentada na Equação (2.9), na qual, $o_0(x)$ é a amplitude da onda de objeto, e ϕ é a fase desta onda.

$$o(x) = o_0(x)e^{-j\phi} \tag{2.9}$$

Com essas ondas definidas, é possível caracterizar o processo de gravação do holograma ao definir a intensidade do padrão de interferência entre as ondas, como feito nas equações (2.10) - (2.12).

$$I(x) = |O(x) + R(x)|^2 \tag{2.10}$$

$$I(x) = r_0^2(x) + o_0^2(x) + r_0(x)o_0(x)\{e^{j[kxsen(\theta_r)+\phi]} + e^{-j[kxsen(\theta_r)+\phi]}\} \tag{2.11}$$

$$I(x) = r_0^2(x) + o_0^2(x) + 2r_0(x)o_0(x)[cos(kxsen(\theta_r) + \phi)] \tag{2.12}$$

O padrão de interferência definido em 2.12 consiste em um conjunto das chamadas franjas por HARIHARAN (2007), com um espaçamento médio entre elas de $ksen(\theta)/2\pi$. Estas franjas tem sua intensidade definida pela amplitude da interferência, e o espaçamento definido pela fase da onda de objeto.

Já para o equacionamento do processo de reconstrução da imagem, é preciso primeiramente definir a transmitância t do filme onde o holograma é gravado. Para tal, é assumido que sua transmitância varia linearmente com seu tempo de exposição a onda de referência, como apresentado na Equação (2.13), onde β_t é uma constante de proporcionalidade, t_0 é a transmitância do filme antes deste ser exposto ao holograma, e T_t é o tempo de exposição.

$$t(x) = t_0 + \beta_t T_t I(x) \tag{2.13}$$

A amplitude de transmitância do holograma é então dado pela Equação (2.14).

$$t(x) = t_0 + \beta_t T_t [r_0^2(x) + o_0^2(x) + r_0(x)o_0(x)\{e^{j[kxsen(\theta_r)+\phi]} + e^{-j[kxsen(\theta_r)+\phi]}\}] \tag{2.14}$$

Com isso é possível definir o processo de reconstrução da imagem como apresentado na Equação (2.16). Desta equação, é possível se tirar quatro termos, onde o primeiro termo corresponde a onda de referência transmitida, o segundo termo é simplesmente uma espécie de auréola que acompanha a onda de referência, o terceiro termo é o que gera a imagem virtual do objeto e o quarto termo é o que gera a imagem conjugada do objeto (HARIHARAN, 2007).

$$U(x) = R(x)t(x) \tag{2.15}$$

$$\begin{aligned} U(x) = (t_0 + \beta_t T_t r_0(x)^2) r_0(x) e^{jkxsen(\theta_r)} \\ + \beta_t T_t r_0(x) o_0^2(x) e^{jkxsen(\theta_r)} \\ + \beta_t T_t r_0^2(x) o(x) \\ + \beta_t T_t r_0^2(x) o^*(x) e^{2jkxsen(\theta_r)} \end{aligned} \tag{2.16}$$

2.4 Metasuperfícies

Metasuperfícies são camadas finas de metamateriais. Em aplicações de eletrônica de micro-ondas e antenas, é geralmente constituída por um arranjo periódico de pequenos elementos metálicos (em termos do comprimento de onda) sobre um meio dielétrico, que são designados de maneira a obter características específicas, geralmente inalcançáveis de outra maneira (FAENZI et al., 2019) (MENCAGLI; MARTINI; MACI, 2015).

Metasuperfícies estão se tornando cada vez mais importante para aplicações na área de antenas e circuitos de micro-ondas. Já que, devido às pequenas dimensões dos elementos que a compõem (geralmente entre $\lambda/10$ e $\lambda/5$), podem ser dispostos milhares de metamateriais sobre um material dielétrico, de maneira a atingir uma impedância de superfície específica de maneira bastante precisa (FAENZI et al., 2019) (MENCAGLI; MARTINI; MACI, 2015). Além disso, é possível modelar estes metamateriais de diversas maneiras, como apresentado por FAENZI et al. (2019), é possível escolher entre diversos formatos para os metamateriais, onde cada um faz com que seja possível atingir determinadas características na modelagem da metasuperfície.

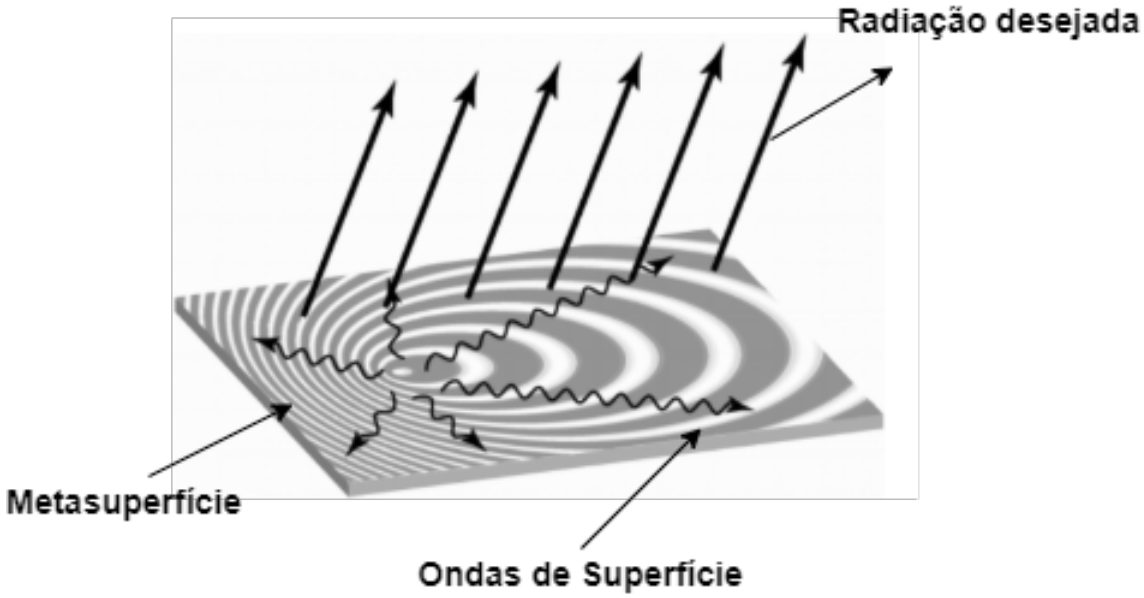
Tais fatores tornam possível modelar uma antena com características superiores à antenas convencionais, seja ao conseguir uma maior diretividade e ganho, ou múltiplos feixes, sem o uso de circuitos ativos complexos, ou múltiplas antenas, mas apenas com circuito impresso, como presente nos vários casos mostrados por FAENZI et al. (2019).

Metasuperfícies são usadas juntamente com técnicas de holografia de modo a conseguir encontrar como devem estar dispostos os metamateriais na metasuperfície. Tal técnica é apresentada e detalhada por FONG et al. (2010), que realiza um processo onde, através de uma onda de superfície associada à onda de referência da holografia ótica, que é utilizada na alimentação da antena, e ao utilizar a conhecida onda emitida necessária para gerar um feixe em determinada direção, análogo à onda de objeto da holografia ótica, é possível encontrar o padrão de interferência entre estas duas ondas, encontrando assim o holograma. Com o holograma definido, estes desenvolvem a metasuperfície ao posicionar metamateriais sobre um material dielétrico, de maneira a conseguir uma impedância artificial com as mesmas características do holograma calculado. Devido as pequenas dimensões dos metamateriais, é possível realizar este procedimento de maneira bastante precisa.

Com a metasuperfície desenvolvida, é possível alimentá-la com a mesma onda de referência utilizada no cálculo do holograma, de modo a conseguir emitir a onda desejada. Este processo está apresentado na Figura 6, onde as setas onduladas indicam as ondas de superfície que alimentam a metasuperfície, as setas retas indicam a radiação desejada utilizada no cálculo do holograma, e a metasuperfície indica a superfície sobre a qual foram posicionados os metamateriais de modo a conseguir uma impedância artificial semelhantes

ao holograma calculado.

Figura 6 – Processo de emissão de feixe por uma metasuperfície



Fonte: FONG et al. (2010)

Técnica semelhante ao utilizado por YURDUSEVEN et al. (2017), com a diferença que este utiliza elementos ativos para modificar o padrão da superfície, de modo a conseguir um apontamento de feixe para diferentes direções, apenas modificando quais elementos da antena devem irradiar ao mesmo tempo.

2.5 Redes Neurais Convolucionais

Uma rede neural convolucional (CNN, do inglês *convolutional neural network*), se trata de um tipo de rede neural especializada para tratar dados com um tipo conhecido de topologia do tipo matricial, como por exemplo, imagens (GOODFELLOW; BENGIO; COURVILLE, 2016). Dada essa especialização das CNN, ela se apresenta como uma boa alternativa para o tratamento de dados como o padrão de radiação de uma antena representado em termos de coordenadas espaciais.

Uma rede neural é desenvolvida com o principal intuito de identificar padrões em dados introduzidos em sua entrada, de modo que, consiga identificar alguma característica pertinente a estes dados. Por exemplo, se uma CNN é treinada para identificar animais presentes em imagens, e esta receba em sua entrada uma imagem de um cachorro, espera-se que a rede neural consiga identificar que há um cachorro presente nesta imagem.

Quando desenvolvendo uma CNN, antes que esta consiga identificar os padrões em um dado introduzido em sua entrada, é necessário, primeiramente, determinar sua arquitetura, que será melhor explicada adiante, e então realizar o processo chamado de treinamento. Seu treinamento consiste em aplicar uma grande quantidade de dados, juntamente com rótulos, que representa alguma característica deste dado, a rede neural, de modo que esta consiga identificar quais padrões são característicos de cada um desses rótulos. Assim, caso seja introduzido, em sua entrada, um dado que possui características pertencentes a algum rótulo presente na base de dados utilizada no processo de treinamento, e se a arquitetura e o treinamento da CNN forem adequados ao problema, esta consegue identificar que este rótulo é relacionado a esse dado. Para o caso onde os dados são imagens, a CNN pode identificar diversos fatores, como, por exemplo, espécies de animais, números, e diversos outros padrões (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.5.1 Operações Realizadas por uma CNN

Uma CNN completa é composta por camadas que realizam operações matemáticas, sendo as mais comumente usadas para extração de características: a convolução, a unidade linear retificada (ReLU, do inglês *rectified linear unit*), a camada de *pooling*. E para a seção de classificação, é utilizada uma camada de achatamento (responsável por converter os dados de um formato matricial, para um formato vetorial), seguida de uma camada totalmente conectada, onde esta identifica padrões lineares e não lineares utilizando um baixo consumo computacional, visto que as características das imagens já foram extraídas nas camadas anteriores da CNN. Por fim, os dados passam por uma camada de ativação *softmax*, responsável por identificar as probabilidades da entrada estar em cada uma das classes possíveis (no caso, cada classe é referente a um rótulo utilizado durante o processo

de treinamento da CNN) (BISWAL, 2021).

2.5.1.1 Convolução

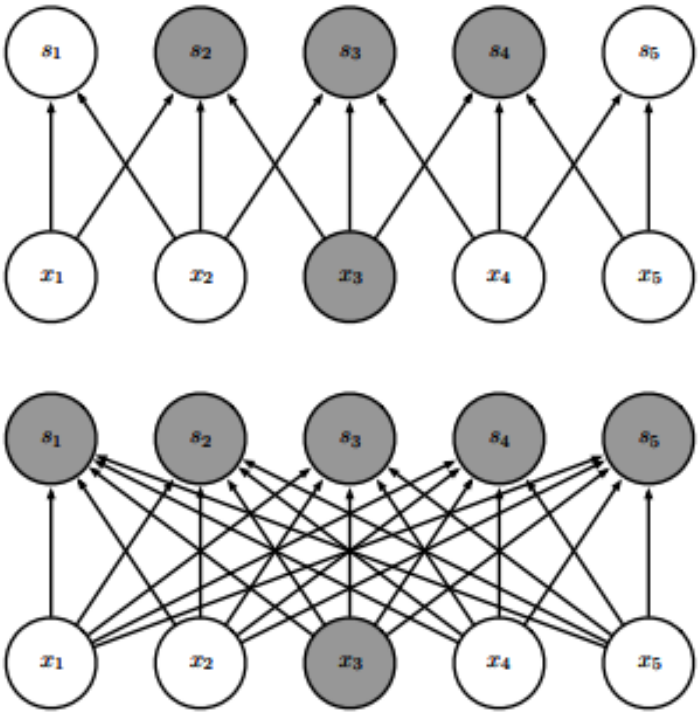
Como o próprio nome sugere, uma operação fundamental no desenvolvimento de redes neurais convolucionais é a convolução. O equacionamento da aplicação da convolução entre uma imagem bidimensional I, de dimensões $N \times M$, e um núcleo de convolução K, de dimensões $L \times L$ é apresentado na Equação (2.17), na qual, S é a imagem filtrada e o sinal $*$ representa a operação de convolução.

$$S(i,j) = K * I = \sum_{g=0}^{L-1} \sum_{h=0}^{L-1} I(i-g, j-h) K(g,h) \tag{2.17}$$

Esta operação é aplicada a sistemas de aprendizado de máquina por conseguir alavancar três importantes ideias, que são interações esparsas, compartilhamento de parâmetros e representações equivariantes, além de tornar possível trabalhar com entradas de diferentes tamanhos (GOODFELLOW; BENGIO; COURVILLE, 2016).

Interações esparsas trazem grande vantagem quando se trabalha com imagens de milhares ou milhões de pixeis, pois é possível separar importantes características da imagem como bordas em núcleos que ocupam um tamanho bastante reduzido, aumentando a eficiência do sistema como um todo. Um exemplo disso pode ser visto na Figura 7, onde ao ser aplicada uma convolução com núcleo de largura três, é possível ver na parte superior da figura onde está destacada a entrada x_3, e como ela só influencia três das saídas, s_2, s_3 e s_4, enquanto na parte de baixo da figura, onde não há conectividade esparsa, cada uma das entradas ocupa cada uma das saídas.

Figura 7 – Exemplo de aplicação de conectividade esparsa, quando aplicada uma convolução com um núcleo de largura três



Fonte: GOODFELLOW, BENGIO e COURVILLE (2016)

Já o compartilhamento de parâmetros indica que é possível durante o aprendizado utilizar um conjunto de parâmetros de uma só vez, aplicados a um único peso, ao invés de utilizar um parâmetro para cada local da imagem, o que diminui o consumo de memória necessário par o armazenamento do modelo.

Devido ao compartilhamento de parâmetros, esta camada da rede neural tem uma propriedade chamada equivariância ao deslocamento, o que indica que se uma imagem for deslocada em x ou y, antes de ser aplicada a convolução, sua saída seria a mesma se fosse aplicada a convolução a imagem inicial, e só então fosse aplicado o deslocamento (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.5.1.2 ReLU

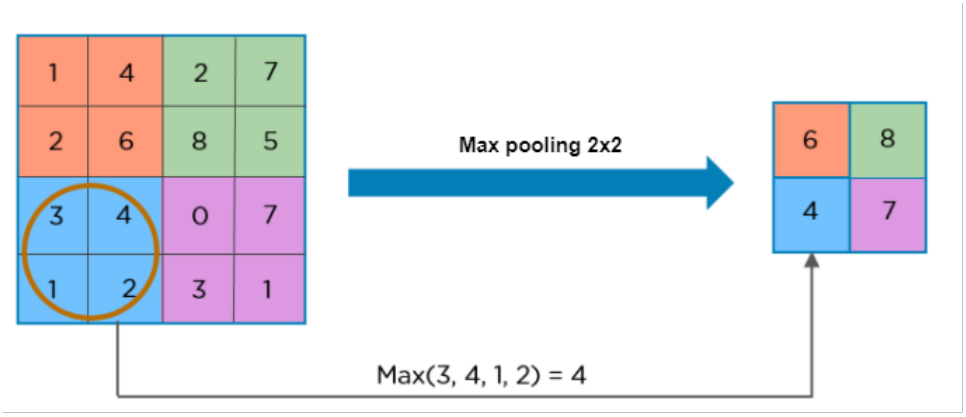
Esta camada da rede neural convolucional se trata de um estágio de ativação, no qual, basicamente, para todos os componentes negativos obtidos na saída da convolução são atribuídos o valor 0, e aos demais são mantidos seus valores originais (BISWAL, 2021). O equacionamento desta operação está apresentando em (2.18), onde x representa a entrada desta função.

$$f(x) = x^+ = max(0, x) \tag{2.18}$$

2.5.1.3 Pooling

Esta camada é responsável por fazer um processo de *downsample*, que reduz as dimensões dos mapas de dados obtidos na saída do estágio de ativação. Um exemplo comum é o uso de filtros de dimensão MxN de *max pooling*, que quando aplicados a uma matriz de dimensão PxQ, esta é dividida em submatrizes de dimensão MxN, das quais são extraídos os valores máximos destas submatrizes. Esta operação identifica padrões importantes em imagens, como por exemplo bordas (GOODFELLOW; BENGIO; COURVILLE, 2016). Um exemplo desta operação pode ser vista na Figura 8, onde um filtro de *max pooling* de dimensão 2x2, é aplicado a uma matriz de dimensão 4x4 (à esquerda da figura), resultando em uma matriz na saída de dimensões 2x2 (à direita da figura), contendo os valores máximos de cada uma das submatrizes. Existem ainda filtros como o *average pooling*, de funcionamento parecido, porém ao invés de utilizar o valor máximo dessas submatrizes, obtém a média destas.

Figura 8 – Exemplo de aplicação do processo de *max pooling* com dimensão 2x2 a uma matriz 4x4



Fonte: BISWAL (2021)

2.5.2 Camada Totalmente Conectada

Existem diversas maneira de se definir a arquitetura de uma CNN, pois esta depende da aplicação par a qual a CNN é desenvolvida. Sendo assim, é possível organizar de diversas maneiras a sequência e quantidade de camadas de uma CNN. Porém, em se tratando da camada totalmente conectada, esta costuma ser aplicada após uma camada de achatamento, que é aplicada após a última camada de *pooling* ou convolução (KHAN et al., 2018). Esta camada de achatamento é responsável por transformar os dados de uma matriz, em um vetor, ou seja, os dados com dimensões $M \times N \times O$, obtidos após a última camada de *pooling*, ou convolução, são transformados em um vetor com dimensão $1 \times (N \cdot M \cdot O)$.

Com os dados preparados pela camada de achatamento, estes são introduzidos em uma camada totalmente conectada, que nada mais é que uma rede neural artificial, que

realiza a operação presente na Equação (2.19) (KHAN et al., 2018). Nesta equação, $f(\cdot)$ é a função de ativação, não linear, para a qual geralmente é utilizada a função ReLU, x é o vetor de entrada obtido após a camada de achatamento, $\mathbf{W}$ representa a matriz contendo os pesos necessários para identificar determinados padrões nas imagens, b é o vetor de tendências, que nada mais é que um vetor que auxilia o processo de identificar os padrões nas imagens, este vetor não necessariamente é inicializado em todas as CNNs, e pode ser usado como sendo um vetor de zeros (VERMA, 2021).

$$y = f(\mathbf{W}x + b) \tag{2.19}$$

É importante frisar que $\mathbf{W}$ possui dimensão $V \times C$, onde V representa a dimensão do vetor obtido na saída da camada de achatamento (no caso do exemplo citado, teria dimensão $N \cdot M \cdot O$), e C representa a quantidade de classes possíveis para as quais as imagens podem ser pertencer. Portanto, y obtido após a camada totalmente conectada, possui dimensão $1 \times C$.

2.5.3 Softmax

A camada de ativação *softmax* é responsável por identificar as probabilidades da imagem aplicada na entrada pertencer a cada uma das classes. E, geralmente, é aplicada após a camada totalmente conectada. Esta função é aplicada através da Equação (2.20), onde C é a quantidade de classes possíveis para as quais as imagens podem pertencer, y representa o vetor obtido após a camada totalmente conectada e $i = 1, 2, ..., C$ (GOODFELLOW; BENGIO; COURVILLE, 2016).

$$softmax(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{C} e^{y_j}} \tag{2.20}$$

A aplicação da camada de ativação *softmax* ao invés de apenas uma operação mais simples se faz necessária, pois, dentro dos valores obtidos na saída da camada totalmente conectada, encontram-se valores negativos, o que resultaria em probabilidades negativas, que são inexistentes, além do fato de serem obtidos valores acima de 1, que também resultaria em uma probabilidade inexistente. A função *softmax* garante, então, que cada componente do vetor de probabilidades obtidos na saída da função *softmax* esteja no intervalo $[0, 1]$ e que a soma de todas as probabilidades somadas seja exatamente 1 (GOODFELLOW; BENGIO; COURVILLE, 2016).

Portanto, a saída da função *softmax* é um vetor contendo as probabilidades da imagem aplicada na entrada da CNN pertencer a uma das classes possíveis. E, no geral, sua previsão é considerada como sendo a classe para a qual é identificada uma maior probabilidade da imagem pertencer.

2.6 Otimização Convexa

Otimização convexa, é um problema na forma da Equação (2.21) (BOYD; VANDENBERGHE, 2004).

$$\text{Minimizar} \quad f_0(x) \tag{2.21a}$$

$$\text{Sujeito a} \quad f_i(x) \leq b_i, i = 1, ..., m \tag{2.21b}$$

Este é considerado convexo, desde que $f_0, ..., f_m : \mathbb{R}^n \to \mathbb{R}$ sejam convexas. Ou seja, satisfazem a condição da Equação (2.22), para todo x, $y \in \mathbb{R}^n$ e todo α, $\beta \in \mathbb{R}$, com $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$. Caso o problema satisfaça essa condição, é possível atingir um mínimo global para sua solução ao resolver o problema de otimização convexa.

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \tag{2.22}$$

Apesar de, no geral, não existir uma forma analítica de resolver problemas de otimização convexa, existem métodos bastante efetivos para resolvê-los, atingindo boas precisões, com um número de iterações que dependem de cada problema sendo tratado (BHATTACHARYYA, 2006).

2.6.1 Otimização Convexa Sobre o Fator de Arranjo

O problema a ser otimizado neste trabalho, se trata do padrão de radiação da antena. E pode ser descrito através da equação do fator de arranjo da antena, que se adaptado para antena desenvolvida neste trabalho, com sua excitação controlada de maneira binária por diodos PIN, possui sua representação como apresentado na Equação (2.23). Onde, $\vec{H}(\vec{p}_{mn})$ é o arranjo de fases do campo magnético na antena, nos pontos onde estão posicionados os elementos irradiantes, $\vec{p}_{mn}$ é o vetor de posições dos elementos irradiantes na antena, $X_{M\times N}$ é a matriz que representa o estado dos elementos irradiantes, no caso, uma matriz binária, indicando a antena desligada, ou ligada.

$$AF(\theta, \phi) = \sum_{m=1}^{M} \left(\sum_{n=1}^{N} X_{M\times N} e^{j\vec{H}(\vec{p}_{mn})} e^{jk(x_{mn} sen(\theta) cos(\phi) + y_{mn} sen(\theta) sen(\phi))} \right) \tag{2.23}$$

O processo de otimização, deve ser, aplicado de modo a otimizar $X_{M\times N}$, identificando assim, quais antenas contribuem para o fator de arranjo desejado, de modo a deixá-las ligadas, e quais não contribuem, de modo a deixá-las desligadas.

Para realizar a solução deste problema, um método foi abordado em (FUCHS; RONDINEAU, 2016), onde foram aplicadas as normas ℓ_1 e ℓ_∞, que são dois extremos,

para regular a solução da otimização. Se combinadas apropriadamente, é possível utilizr essas normas de modo a forçar a solução da otimização convexa a ser binária, algo que não ocorre, se for utilizada apenas uma das normas, por exemplo. O procedimento aplicado por FUCHS e RONDINEAU (2016), está apresentado na Equação (2.24), na qual α é um escalar utilizado para limitar os possíveis valores da solução, para o problema em questão, se $\alpha = 0.5$, os valores são obrigados a ficarem entre 0 e 1 e o parâmetro γ é responsável por controlar como é o comportamento da solução, se ele leva para uma matriz com mais elementos 0 ou 1, e deve ser ajustado de acordo com o problema abordado. Uma solução possível para encontrar o valor de γ é a realização de diversos processos de otimização computacionalmente, para diferentes valores de γ, e identificar qual valor melhor se adéqua ao problema (FUCHS; RONDINEAU, 2016).

$$\text{Minimizar} \quad ||X_{M\times N}||_1 + \gamma||X_{M\times N} - \alpha||_\infty \tag{2.24a}$$

$$\text{Sujeito a} \quad |AF(\theta,\phi) - D_f(\theta,\phi)| \ \leq \varepsilon(\theta,\phi) \tag{2.24b}$$

A expressão $D_f(\theta,\phi)$ representa o padrão de radiação desejado, e $\varepsilon(\theta,\phi)$ é a margem de erro considerada para o problema, ou seja, o nível dos lóbulos secundários aceitável para o problema.

Para realizar esta solução, é necessário um ponto de partida para identificar um valor inicial do fator de arranjo, e da matriz de estados dos elementos irradiantes. Para tal, é utilizado o método da retro-propagação. Entretanto, para incorporar este ponto de partida ao problema, não é possível otimizar estes dados diretamente, fazendo-se necessário o uso de uma variável auxiliar, assim como no procedimento realizado por SANTANA (2021). Esta variável auxiliar está apresentada na Equação (2.25), na qual, $U_{M\times N}$ é uma matriz que possui todos os elementos iguais a 1, $H_{M\times N}$ é a matriz de estados dos elementos irradiantes obtida pelo método da retro-propagação, $\circ$ representa o produto de Hadamard e $A_{M\times N}$ é a matriz auxiliar.

$$A_{M\times N} = H_{M\times N} + |(H_{M\times N} - U_{M\times N})| \circ X_{M\times N} \tag{2.25}$$

Após o problema de otimização ser solucionado, a variável auxiliar $A_{M\times N}$ assume o papel de $X_{M\times N}$ na Equação (2.23), referente ao fator de arranjo da antena, o que gera a otimização no apontamento do feixe.

3 Desenvolvimento da Antena

3.1 Antena de Metasuperfície

A composição de uma antena de metasuperfície, consiste em uma fonte de alimentação, um meio propagante, e elementos irradiantes. Seus elementos irradiantes podem conter ou não um componente de controle, sendo que, caso os elementos possuam este componente de controle, se torna possível a reconfiguração desta antena de modo a permitir a antena apontar feixes para diversas direções.

Para este projeto, a antena proposta conta com um guia de onda de placas paralelas como meio propagante, um conector coaxial lateral como alimentador, dipolos do tipo fenda como elementos irradiante e diodos pin como componentes de controle.

3.2 Meio Propagante

O meio propagante foi escolhido como sendo uma cavidade composta por um guia de onda de placas paralelas, onde duas camadas de cobre são posicionadas sobre cada uma das faces de um substrato dielétrico.

O substrato escolhido se trata do laminado Kappa438 da fabricante Rogers, cujas principais características para o design da da cavidade estão apresentadas na Tabela 1. Tendo em vista que este foi escolhido por possuir um baixo preço dentre os substratos utilizados na fabricação de antenas, e apesar de ser mais caro que o tradicional FR-4, possui performance e confiabilidade superior a este, contendo assim uma mistura de baixo preço, boa performance e confiabilidade (ROGERS, 2019).

Tabela 1 – Ficha de dados do laminado Kappa438

Característica	Valor	Unidade de Medida
Constante dielétrica	4,38	-
Altura	0,508	mm
Fator de Dissipação (σ)	0,005	-

Fonte: Rogers (2019)

É importante ressaltar que apesar deste substrato ter sido escolhido principalmente pelo baixo custo, ele não possui uma constante dielétrica elevada, o que diminuiria o comprimento de onda guiado, e por consequência o tamanho do elemento irradiante, e caso esse acabasse muito pequeno, poderia resultar em dificuldades na fabricação, já que cada fabricante de circuitos impressos possuem um valor mínimo de comprimento e largura de fendas que eles conseguem fabricar.

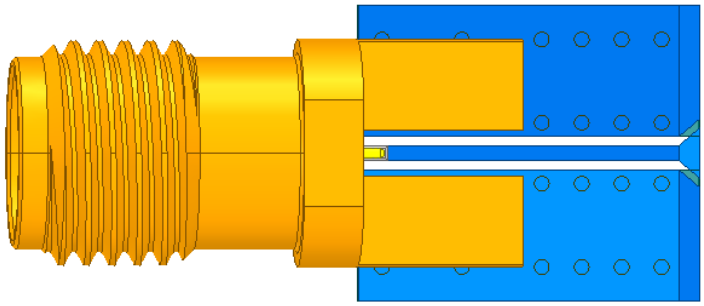
3.3 Alimentador

Para a alimentação da antena foi escolhido um conector coaxial de borda, cuja conexão à antena conta com um guia de ondas coplanares (CPW, de coplanar waveguide). O qual foi escolhido como responsável por lançar a onda de referência no meio propagante, devido ao fato deste poder ser acoplado à antena, para gerar as ondas de referência, sem a necessidade do uso de uma outra antena para iluminar o circuito, como feito por YANG et al. (2016) e TAO et al. (2020).

Outra maneira de alimentar a antena, seria utilizar outro tipo de conector que pode ser fixado à antena, como um conector coaxial monopolo posicionado em seu centro, porém o conector coaxial de borda foi escolhido por apresentar um melhor casamento de impedância para a antena dentre as opções testadas.

O circuito contendo o conector de borda e o guia de ondas coplanares utilizados para conectar este à antena está apresentadona figura 9, é notado também o uso de pinos pela região do guia de ondas, isso é feito para aumentar a eficiência da potência radiada, o que melhora também o casamento de impedância da antena.

Figura 9 – Conector de borda e guia de ondas coplanares utilizados para alimentar a antena



Fonte: Autoria Própria

3.4 Elemento Irradiante

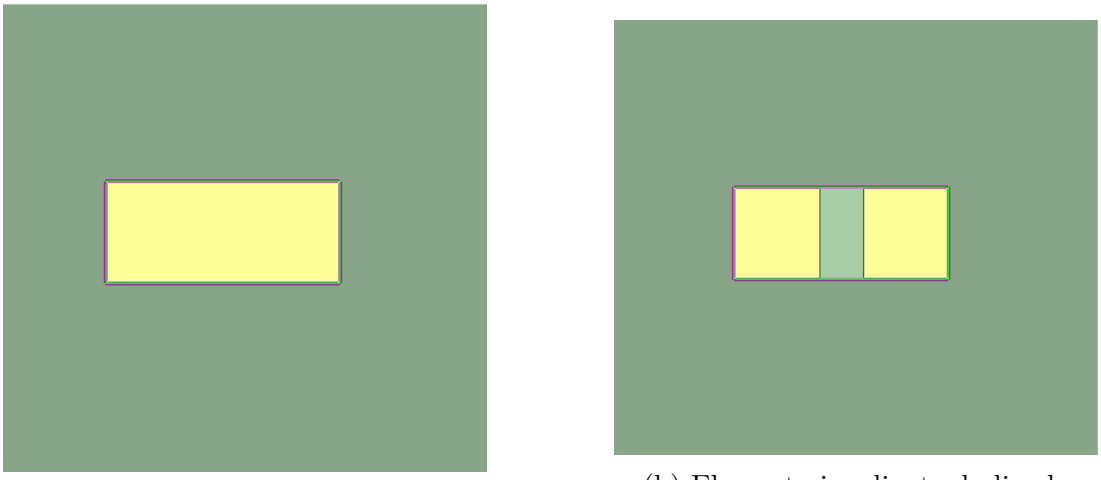
Para a escolha do elemento irradiante para uma metasuperfície, é possível escolher dentre diversos tipos de elementos, cada um com suas próprias características, como apresentado em (FAENZI et al., 2019). Porém, para o caso de antenas reconfiguráveis em metasuperfícies, estes tipos de elementos possuem menos opções se comparado com antenas não reconfiguráveis, e os principais formatos de elementos mais utilizados são apresentado por SHANG et al. (2021),

O elemento escolhido para esta antena em questão, é baseado no elemento apresentado por YURDUSEVEN et al. (2017), que se trata de um dipolo do tipo fenda, controlado

por um chaveador posicionado em seu centro. O chaveador funciona de modo a interligar os dois lados da superfície metálica na qual a fenda está posicionado para controlar a resposta ao acoplamento dos elementos ao modo guiado que o alimenta, modificando assim, sua frequência de ressonância.

Um exemplo sobre o funcionamento do chaveador sobre o elemento irradiante está apresentado na Figura 10. Onde na subfigura 10a está apresentado o elemento original, considerando o chaveador desligado (não permitindo a passagem de corrente), fazendo com que o dipolo esteja em seu modo irradiante (ligado). Enquanto na subfigura 10b, está apresentado o como o elemento estaria caso o chaveador estivesse ligado (permitindo a passagem de corrente), fazendo com que o dipolo tenha um comprimento levemente menor que metade de seu tamanho, fazendo com que o seu acoplamento ao modo guiado que o alimenta mude, fazendo assim com que ele esteja em seu modo não irradiante (desligado).

Figura 10 – Funcionamento do componente chaveador sobre o elemento irradiante



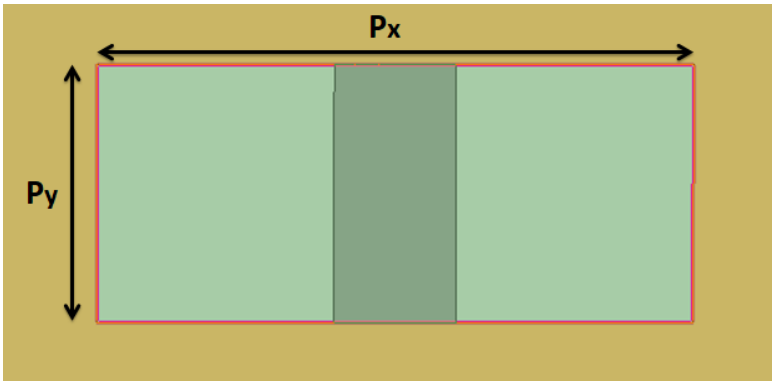
(a) Elemento irradiante ligado

(b) Elemento irradiante desligado

Fonte: Autoria própria.

Para o projeto em questão, o elemento irradiante desenvolvido está apresentado na Figura 11, onde P_x indica o comprimento do dipolo, cujo valor ideal para acoplamento da onda guiada deve ser de $\lambda_g/2$. Porém, apesar deste comprimento trazer um máximo acoplamento do modo guiado, isto introduz perturbações no modo guiado, e portanto, foi utilizada uma fenda com comprimento de $\lambda_g/3.5$. O valor de P_y não representa grandes implicações na frequência sobre o qual este trabalha, e portanto pode ser otimizado de modo a proporcionar melhores resultados ao projeto, a depender do que é considerado mais importante, como por exemplo o ganho, ou o casamento de impedância, o melhor valor encontrado dentre os testados foi o valor de $\lambda_g/8$.

Figura 11 – Antena dipolo do tipo fenda desenvolvida



Fonte: Autoria Própria

Na Figura 11 é possível ver ainda, que ao centro do dipolo está posicionado um retângulo. Este representa o componente de controle, porém, como este projeto encontra-se ainda incompleto, ao invés do uso de um componente comercial, está sendo usado uma estrutura que utiliza a condição de fronteira de condutor elétrico perfeito (PEC, de perfect electric conductor) para simular um chaveador. Esta condição de fronteira está presente no software em que a antena foi desenvolvida, o High Frequency Structure Simulator (HFSS) da empresa Ansys. Caso adicionada a condição de PEC a uma estrutura sem material definido no software HFSS, este se comporta como um condutor elétrico perfeito, simulando assim um circuito fechado, e caso não seja adicionada nenhuma condição de fronteira a esta estrutura, ela não conduz corrente e se comporta como um circuito aberto. Simulando assim, o comportamento de um chaveador, ligando e desligando a depender da condição de fronteira adicionada a ele.

3.5 Componente de Controle

Embora ainda não tenha sido aplicado ao design da antena, foi escolhido já foi escolhido o componente chaveador a ser utilizado. Para o controle deste tipo de antena, é possível selecionar dentre vários tipos de chaveadores de radiofrequência (RF). Para encontrar a melhor opção, é possível olhar uma tabela de comparação desenvolvida pela empresa de fabricante de equipamentos e softwares de simulação em RF KEYSIGHT (2017), que avalia as características de cada componente, considerando as médias do mercado.

Desta tabela, foram separadas as principais características para este sistema, e tal comparação está apresentada na Tabela 2, que leva em consideração os principais fatores para um projeto em alta frequência, que são sua perda de inserção (IL, de insertion loss), que representam as perdas do sistema quando o sistema deveria conduzir, sua isolação, que indica a capacidade do diodo de conter uma corrente reversa, estando esta dividida

em alta frequência (AF) e baixa frequência (BF), além da comparação de velocidade de chaveamento e consumo de potência.

Tabela 2 – Comparação entre os possíveis componentes de controle para a antena selecionada

Parâmetro	**Diodo Pin**	**FET**	**Híbridos**	**MEMS**
IL	MÉDIA	ALTA	ALTA	BAIXA
Isolação	BOA (AF)	BOA (BF)	BOA (AF)	BOA
Velocidade de chaveamento	RÁPIDO	MÉDIA	MÉDIA	LENTO
Consumo de potência	ALTO	BAIXO	MODERADO	BAIXO

Fonte: KEYSIGHT (2017)

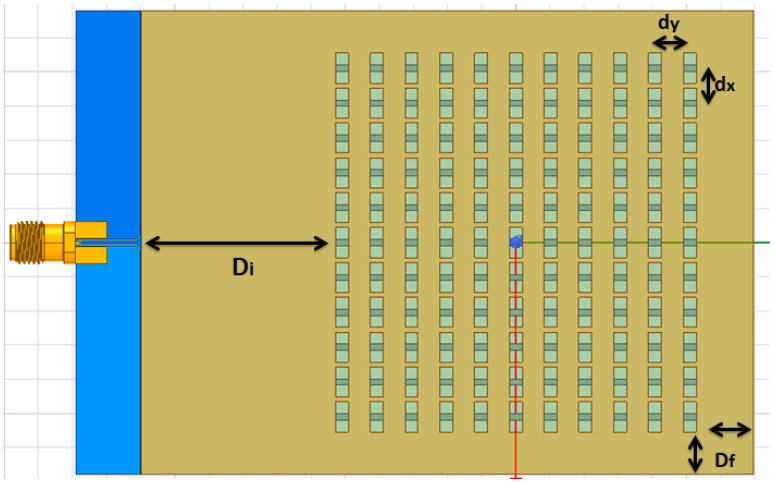
Levando isso em conta, foi escolhido o uso de diodos PIN como componente chaveador, tendo em vista que este apresenta uma perda de inserção média, com uma boa isolação, que são os principais fatores, no que diz respeito ao controle de passagem de corrente por este. Sendo escolhido ao invés de RF-MEMS, devido à grande diferença na velocidade de chaveamento entre estes, e a diferença no consumo de potência não é tão relevante para sistemas de comunicação localizados em solo. Já a velocidade de chaveamento pode ser bastante influente para o caso de um sistema de comunicação, visto que, caso uma antena demore a mudar o apontamento de feixe para a direção necessária para captar ou enviar sinais para um satélite, por exemplo, isso resulta em uma perda de sinal, o que nunca é desejado neste tipo de sistema.

3.6 Integração da Antena

Com tudo isso definido, é possível montar a antena completa, contendo um arranjo de antenas dipolo do tipo fenda dispostos sobre a cavidade, já com a fonte de alimentação responsável por introduzir a onda de referência à metasuperfície. As Figuras 12-14 mostram a vista superior, a inferior e a isométrica de uma das antenas desenvolvidas, de modo a permitir uma boa observação do material desenvolvido.

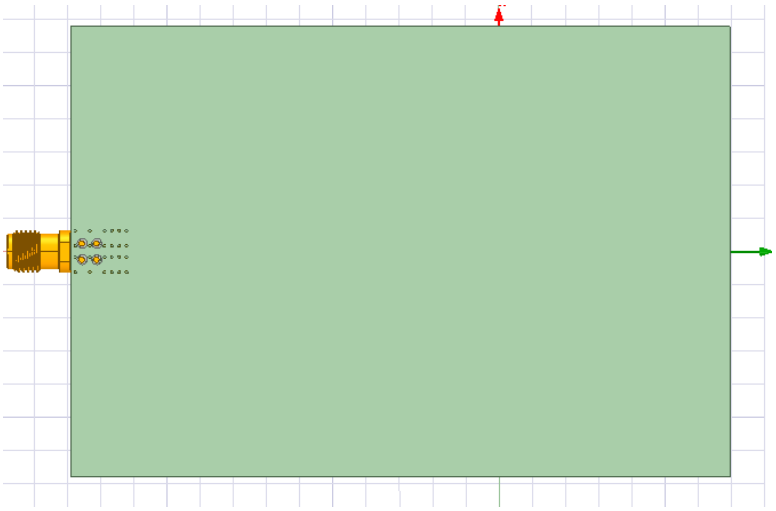
Na Figura 12, é possível ver como os slots são distribuídos periodicamente pela cavidade, formando um arranjo de 11x11 elementos, com distância entre os elementos de $d_x = d_y = \lambda_g/3$. Além da área contendo os elementos, há também um espaço entre o conector e a região onde encontram-se os elementos irradiantes, sendo essa distância de $D_f = \lambda$, além de uma distância entre os elementos e as bordas da antena de $D_i = \lambda/5$, essas distâncias relativamente grandes se fazem necessárias para que seja possível gerar uma fase melhor distribuída pela cavidade, o que gera uma melhor capacidade de apontamento para a antena, e para tal é necessário mais espaço para as ondas eletromagnéticas se propagarem.

Figura 12 – Vista superior da antena



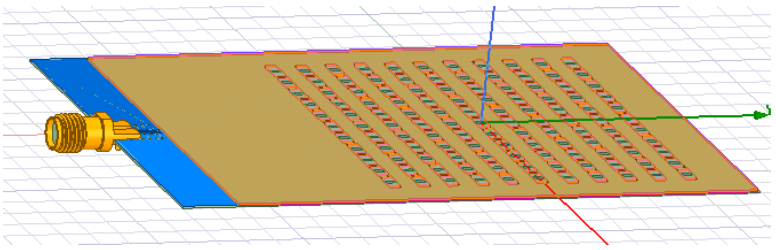
Fonte: Autoria Própria

Figura 13 – Vista inferior da antena



Fonte: Autoria Própria

Figura 14 – Vista isométrica da antena

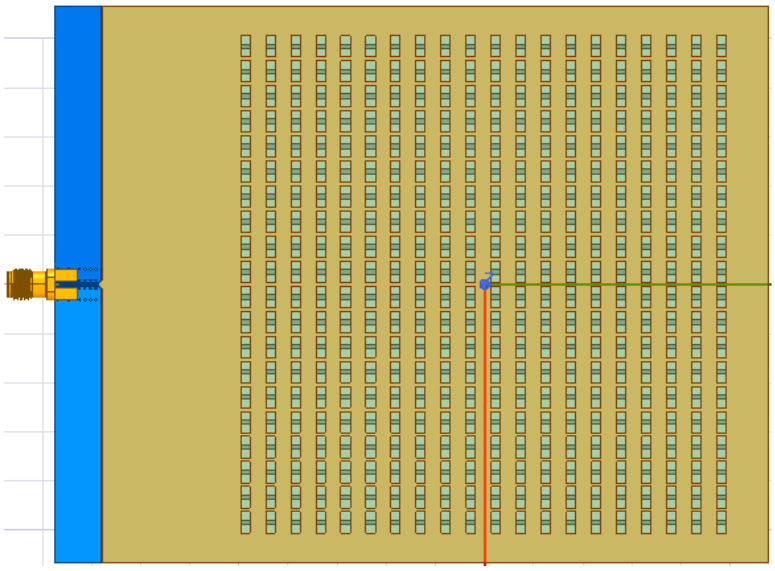


Fonte: Autoria Própria

Para que fosse possível uma comparação entre os métodos de otimização de feixe utilizados em diferentes situações, foi desenvolvida ainda, uma segunda antena. Esta contendo 20x20 elementos irradiantes. Mantendo as demais características iguais a antena

com 11x11 elementos, sendo o número de elementos, e por consequência, o tamanho total da antena, as únicas diferenças. A Figura 15, apresenta a vista superior desta segunda antena.

Figura 15 – Vista superior da antena com 20x20 elementos



Fonte: Autoria Própria

3.6.1 Dimensões da Antena

Considerando os dados referentes a frequência de operação da antena (12GHz), e os dados do substrato apresentado na tabela 1, é possível encontrar as dimensões já especificadas ao longo do documento, porém em unidades de medida do sistema internacional de medidas, visto que estas estavam apresentadas em termos de comprimento de onda e comprimento de onda guiado. Os comprimentos calculados estão apresentados na Tabela 3.

Tabela 3 – Dimensões calculadas para as variáveis de comprimento da antena

Variável	Comprimento [mm]
P_x	4,4
P_y	1,9
d_x	5,1
d_y	5,1
D_i	25
D_f	5,0

Fonte: Autoria própria

3.7 Procedimento Holográfico

A técnica de holografia utilizada neste projeto, se assemelha bastante ao procedimento holográfico utilizado na holografia ótica, possuindo as etapas de encontrar o

holograma, e de reconstrução da onda desejada. Mais especificamente, é realizado o processo primeiramente encontrando qual a distribuição de campo na antena que geraria o padrão de radiação de interesse. Seguido da configuração da metasuperfície que seria capaz de gerar essa distribuição de campos sobre a antena, quando excitado por uma onda de referência, assim como apresentado por YURDUSEVEN et al. (2017).

3.7.1 Cálculos Realizados

Para que seja captado o padrão de interferência no caso do projeto, escolhe-se um ponto como foco em uma posição **r'**(x', y', z') que indica a direção para a qual deseja-se apontar o feixe. E com esse ponto, é encontrado através do método da retro-propagação, como um campo eletromagnético apontado nesta direção, interfere na superfície da antena através da Equação (3.1), onde k representa o número de onda, e **r**(x, y, z) representa os vetores com as posições de cada elemento na superfície da antena.

$$\mathbf{BP} = \frac{e^{-jk(|\mathbf{r}\text{-}\mathbf{r}'|)}}{|\mathbf{r}\text{-}\mathbf{r}'|} \tag{3.1}$$

Dado que para reconstruir a imagem é necessário o padrão de interferência deste feixe sobre a antena, é necessário calcular como essa onda interfere sobre a onda de referência que age sobre os elementos da antena, o cálculo dessa interferência é dado pela Equação (3.2), onde $\mathbf{H}^*_{ref}$ indica o complexo conjugado do campo magnético de referência.

$$\mathbf{I} = \mathbf{BP} \cdot \mathbf{H}^*_{ref} \tag{3.2}$$

Através da Equação (3.2), é visível que em se tratando apenas da fase, o que está acontecendo é uma subtração entre a fase da onda obtida pelo método da retro-propagação, e a onda de referência que age sobre a antena. E como a fase é o que nos interessa, pode-se utilizar apenas a Equação (3.3) para encontrar o padrão de interferência.

$$\mathbf{I}_\phi = \angle\underline{\mathbf{BP}} - \angle\underline{\mathbf{H}_{ref}} \tag{3.3}$$

Por fim, analisa-se o resultado obtido na Equação (3.3), e observa-se para quais elementos da antena o padrão de interferência possui uma fase inferior a um limiar (no caso, o valor utilizado foi de $\pm 25^o$, pois apresentou melhores resultados), entende-se que ele auxilia na formação do feixe desejado, ou seja, para os casos onde essa condição é atendida, deve-se permitir que esse elemento irradie (chaveador desligado), e para os casos onde essa condição não é atendida, não deve-se permitir que o elemento irradie (chaveador ligado), encontrando assim o holograma, que se trata de uma matriz composta por 0's e 1's, que indicam quais antenas devem ou não irradiar.

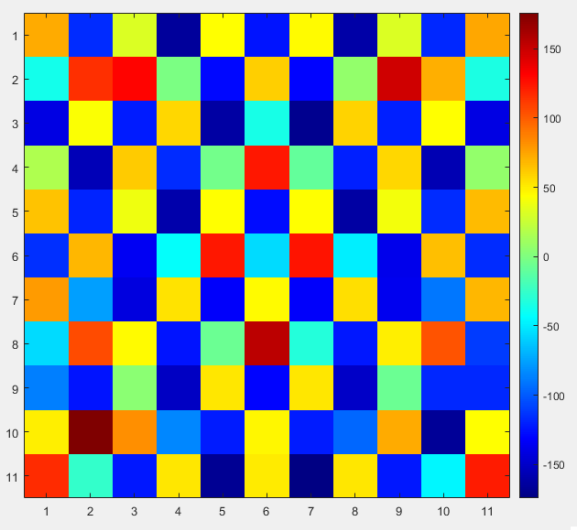
3.7.2 Implementação do Procedimento Holográfico

Antes da implementação do procedimento holográfico, é importante notar que devido a geometria do elemento escolhido, este irá acoplar apenas campos magnéticos no sentido ao qual seu lado de maior dimensão está orientado, no caso o eixo X, o que interfere no cálculo da Equação (3.3), já que, como apenas a componente X do campo magnético de referência é acoplada ao elemento radiante, apenas esta componente da onda de referência que age sobre a antena deve ser usada para o cálculo do holograma.

Para o cálculo do holograma, foi realizada inicialmente uma simulação da metasuperfície, e extraído dela seus campos magnéticos, de modo a se obter os dados da onda de referência que age sobre os elementos irradiantes. Após isso, foi obtida a fase da componente X do campo magnético, para que com ela fosse possível realizar a aplicação, juntamente com a fase obtida pelo método de retro-propagação para cada feixe desejado, na Equação (3.3) e encontrar assim o holograma. É importante ressaltar que todo o desenvolvimento e simulações referentes ao circuito foram realizadas utilizando o software de desenvolvimento HFSS, buscando resultados confiáveis para confirmar o funcionamento do projeto.

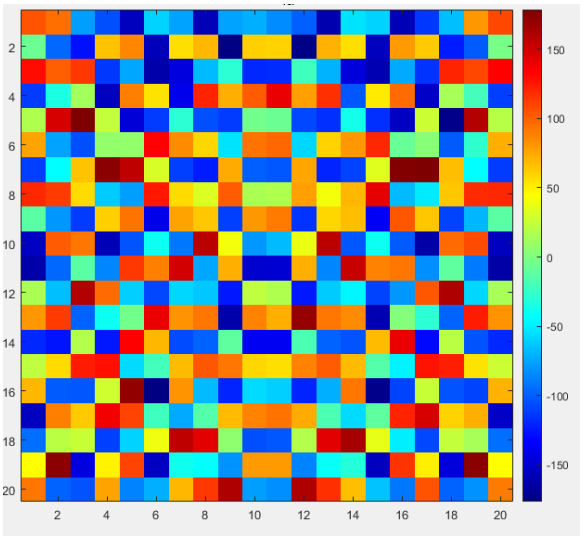
É importante ressaltar, que o o campo magnético obtido, e os valores calculados pelo método da retro propagação são discretizados para as posições da metasuperfície onde encontra-se cada elemento. A distribuição de fase referente à onda de referência simulada no HFSS e discretizada para os elementos irradiantes está apresentada na Figura 16, para a antena com 11x11 elementos e na Figura 17, para a antena com 20x20 elementos.

Figura 16 – Fase de H_x em cada elemento da metasuperfície, da onda de referência, em graus, para a antena com 11x11 elementos.



Fonte: Autoria Própria

Figura 17 – Fase de H_x em cada elemento da metasuperfície, da onda de referência, em graus, para a antena com 20x20 elementos.



Fonte: Autoria Própria

Os códigos desenvolvidos para cálculo do holograma pelo método da retro-propagação está apresentado no Apêndice A, os códigos apresentados neste apêndice, contém ainda, a aplicação da técnica de otimização convexa, e os testes dos hologramas obtidos pela CNN.

4 Desenvolvimento da Rede Neural Convolucional

4.1 Entradas e Saídas da Rede Neural Convolucional

Semelhante ao processo de otimização convexa a ser aplicado a este problema, a CNN desenvolvida, visará uma otimização no fator de arranjo da antena, que por consequência gera uma otimização no feixe apontado. A otimização do fator de arranjo, ao invés da otimização de algum outro fator como o ganho, ou diretividade, se faz necessário, pois a obtenção destes, através de um software de simulação convencional, leva um tempo de simulação bastante elevado, tornando inviável a geração de uma base de dados para o problema.

Para as entradas da CNN, foram utilizadas imagens contendo os fatores de arranjo obtidos como resultado da antena após a aplicação do holograma obtido pelo método da retro-propagação. Mais especificamente, imagens com dimensões 37 x 37, sendo essas dimensões relacionadas aos passos em θ e ϕ utilizados no cálculo do fator de arranjo (ϕ indo de 0 à 2π Com passo de $\pi/18$ e θ indo de 0 à π com passo de $\pi/36$). Sendo estes passos escolhidos de modo a não se obter imagens com dimensões muito grandes, que tornaria inviável sua aplicação na rede neural em um computador pessoal, nem imagens pequenas que resultariam em uma baixa resolução para os resultados calculados.

Já para seu processo de treinamento e previsão, foram usados como rótulos para estas imagens, a notação decimal para cada linha da matriz do holograma. Sendo esta notação utilizada aproveitando o fato do holograma ter um padrão binário, ou seja, os valores de cada elemento da linha, tem valores 0 ou 1. Permitindo assim, a conversão de uma linha de N componentes (11 para a primeira antena e 20 para a segunda antena) em um número decimal indo de 0 a $2^N - 1$. Esta conversão se fez necessária, pois, cerca de 90% dos elementos da matriz permanecem desligados para o apontamento de cada feixe, o que fez com que a CNN durante seu processo de otimização entendesse que seria mais precisa ao definir que cada elemento sempre estava em 0, gerando assim, resultados indesejados. Não é possível, entretanto, utilizar toda a matriz em notação decimal, pois a quantidade de classes possíveis para a CNN identificar (de 0 a $2^{N^2} - 1$) tornavam o funcionamento da CNN muito lento.

Com isto definido, é possível explicar em resumo que a CNN recebe como entrada para seu treinamento as imagens de dimensão 37x37 representando os fatores de arranjo para o apontamento do feixe, e como rótulos, recebe N valores decimais, representando os elementos ligados e desligados de cada linha do arranjo de antenas.

Já na sua etapa de previsão, a CNN recebe como entrada uma imagem do fator de arranjo que representa o apontamento de feixe desejado (um fator de arranjo semelhante ao obtido pelo método da retro-propagação, porém com os lóbulos secundários reduzidos), e dá como saída, os valores decimais, referentes aos elementos que devem ser ligados ou desligados para realizar o apontamento desejado.

4.2 Estrutura da Rede Neural Convolucional

Para a escolha da rede neural convolucional a ser utilizada, foi utilizada uma rede neural convolucional profunda, baseado na rede neural desenvolvida por SIMONYAN e ZISSERMAN (2015), onde estes mostram que a eficiência no reconhecimento de imagens de uma CNN é melhorada ao se adicionar várias camadas de convolução, porém com núcleos de pequenas dimensões (no caso, 3x3), ao invés de se utilizar apenas uma única camada com um filtros de dimensão maior.

Para a CNN desenvolvida para este trabalho, foi levado em consideração que a quantidade de pixels presentes nas imagens era inferior as imagens utilizadas por SIMONYAN e ZISSERMAN (2015), portanto foram utilizados filtros de dimensões ainda menores. Mais especificamente, foi utilizada a mesma técnica utilizada por TAO et al. (2020), onde são adicionados múltiplos filtros convolucionais de dimensão 1x1, que não alteram o tamanho da imagem, mas trazem melhora nos resultados da rede neural.

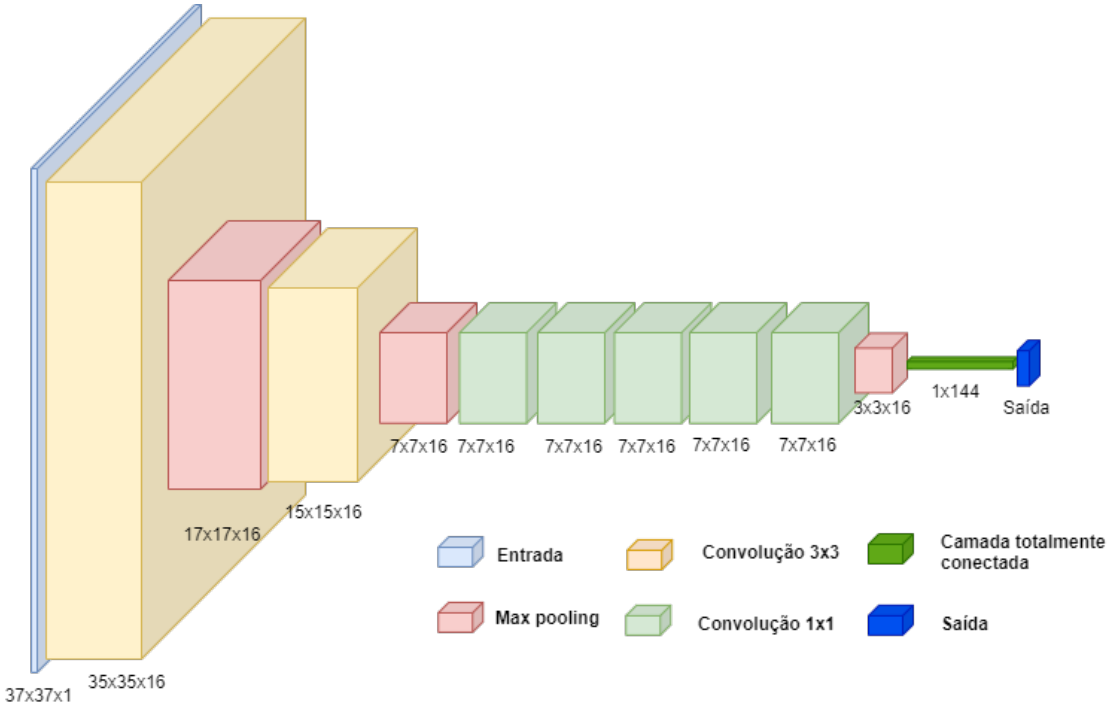
Para ser mais específico sobre a estrutura da CNN, o sumário da rede neural desenvolvida está apresentado na Figura 18, sumário este extraído da CNN que foi desenvolvida utilizando a biblioteca Keras, que se trata de uma biblioteca de redes neurais de código aberto escrita em Python. E na Figura 19, está apresentado o esquemático da arquitetura proposta para a CNN, de modo a facilitar a interpretação das camadas utilizadas, e como os dados são tratados por esta.

Figura 18 – Sumário da CNN desenvolvida

Layer (type)	Output Shape	Param #
conv2d_63 (Conv2D)	(None, 35, 35, 16)	160
max_pooling2d_27 (MaxPooling	(None, 17, 17, 16)	0
conv2d_64 (Conv2D)	(None, 15, 15, 16)	2320
max_pooling2d_28 (MaxPooling	(None, 7, 7, 16)	0
conv2d_65 (Conv2D)	(None, 7, 7, 16)	272
conv2d_66 (Conv2D)	(None, 7, 7, 16)	272
conv2d_67 (Conv2D)	(None, 7, 7, 16)	272
conv2d_68 (Conv2D)	(None, 7, 7, 16)	272
conv2d_69 (Conv2D)	(None, 7, 7, 16)	272
max_pooling2d_29 (MaxPooling	(None, 3, 3, 16)	0
flatten_9 (Flatten)	(None, 144)	0
dense_9 (Dense)	(None, 2048)	296960

Fonte: Autoria Própria

Figura 19 – Esquemático da CNN desenvolvida



Fonte: Autoria Própria

A CNN foi desenvolvida considerando uma imagem na entrada de dimensão 37x37, e foi utilizada uma função de ativação RelU após cada camada. Foram ainda, utilizados 16 filtros por camada para a parte de extração de características, a qual é composta por: uma camada de convolução com núcleo de dimensão 3x3, seguidas por uma camada de max

pooling 2x2, foram adicionadas duas vezes estas camadas. Após estas, foram adicionadas mais cinco camadas de convolução com dimensão 1x1, seguidas de mais uma camada de max pooling 2x2, resultando em uma saída de dimensão 3x3x16.

Já para a parte de classificação, foi utilizada uma camada de achatamento, que para o caso específico, converte os dados de uma matriz 3x3x16, para um vetor com dimensões $5 \cdot 5 \cdot 16$x1 = 144x1. Seguida de uma camada totalmente conectada de dimensão N^2, onde N é o número de elementos irradiantes nas linhas da antena, no caso, 11 para a primeira antena e 20 para a segunda antena. Sendo esta dimensão escolhida por representar o número de classes possíveis para as quais cada entrada pode ser associada. Por fim, para determinar as probabilidades de cada elemento ser associado a cada classe, foi utilizada a função de ativação softmax.

Os códigos desenvolvidos para a geração da base de dados utilizadas para treino e teste da CNN estão dispostos no Apêndice B. Já os códigos desenvolvidos para a CNN estão dispostos no apêndice C, para um maior detalhamento do conteúdo.

5 Resultados e Análises

5.1 Apontamento de Feixe pelo Método da Retro-Propagação

Primeiramente, para realizar um teste sobre o funcionamento da antena desenvolvida, assim como o funcionamento de sua reconfiguração de feixe, foram calculados os hologramas pelo método da retro-propagação para diferentes apontamentos de feixe e realizadas simulações no software HFSS para comprovar sua funcionalidade. Sendo este trabalho realizado sobre a primeira antena, com 11x11 elementos, pois para a segunda antena, o computador pessoal utilizado apresentava grande demora para realização das simulações. Além disso, para o apontamento de certos feixes, a solução não convergiu devido a falta de memória para tal, gerando assim, um resultado não confiável.

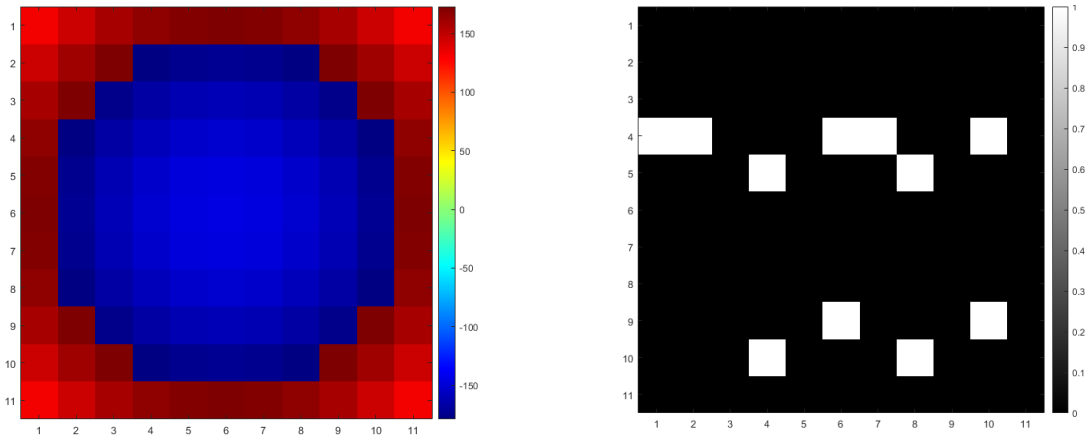
Durante este processo de teste do funcionamento da antena, foram encontrados através de cálculos realizados no software MATLAB, os gráficos representando a fase da onda calculada pelo método da retro propagação, que interfere em cada elemento da antena, assim como os gráficos referentes aos hologramas calculados. Estes estão apresentados nas seções 5.1.1 - 5.1.4 juntamente com os valores da diretividade simulada, para dar um exemplo de como foi realizado todo o processo de holografia.

Os valores simulados dos diagramas de radiação da diretividade estão apresentados tanto em escala linear quanto em dB, pois, o apontamento em algumas direções apresentam resultados de difícil interpretação, se apresentados em dB.

5.1.1 Apontamento de feixe,em coordenadas esféricas, para os ângulos

$\theta = 0, \phi = 0$

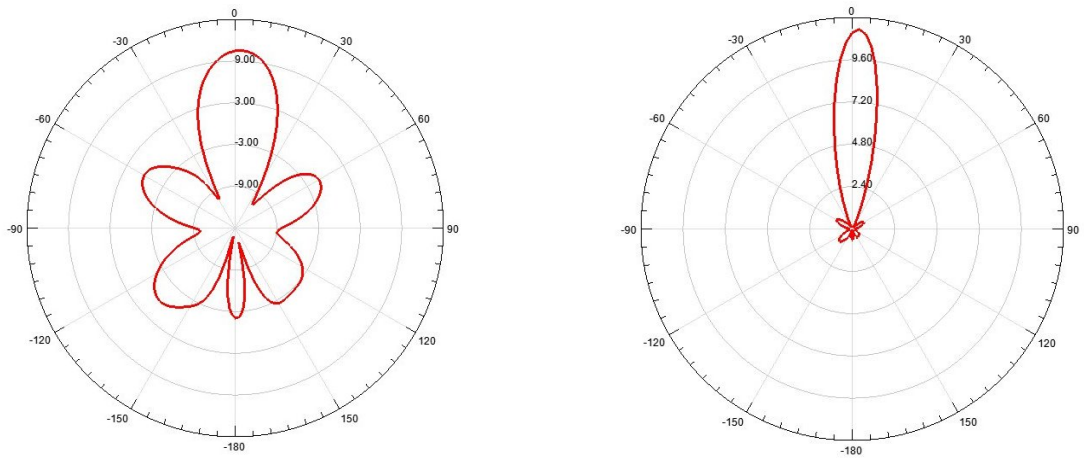
Figura 20 – Cálculo do holograma para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 0$ e $\phi = 0$. (a) Fase obtida através da retro-propagação em cada elemento da metasuperfície em graus. (b) Holograma encontrado do padrão de interferência onde em 1 (branco) são as antenas que devem irradiar e em 0 (preto) são as antenas que não devem irradiar.



(a) (b)

Fonte: Autoria própria.

Figura 21 – Diagramas de radiação simulados no HFSS para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 0$ e $\phi = 0$. (a) Diretividade em dB. (b) Diretividade em escala linear.



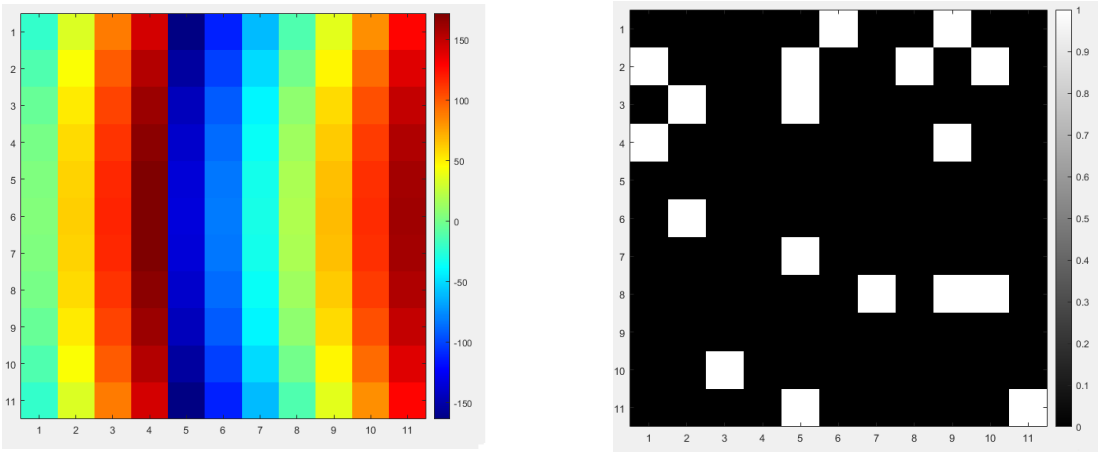
(a) (b)

Fonte: Autoria própria.

5.1.2 Apontamento de feixe,em coordenadas esféricas, para os ângulos

$\theta = 45, \phi = 90$

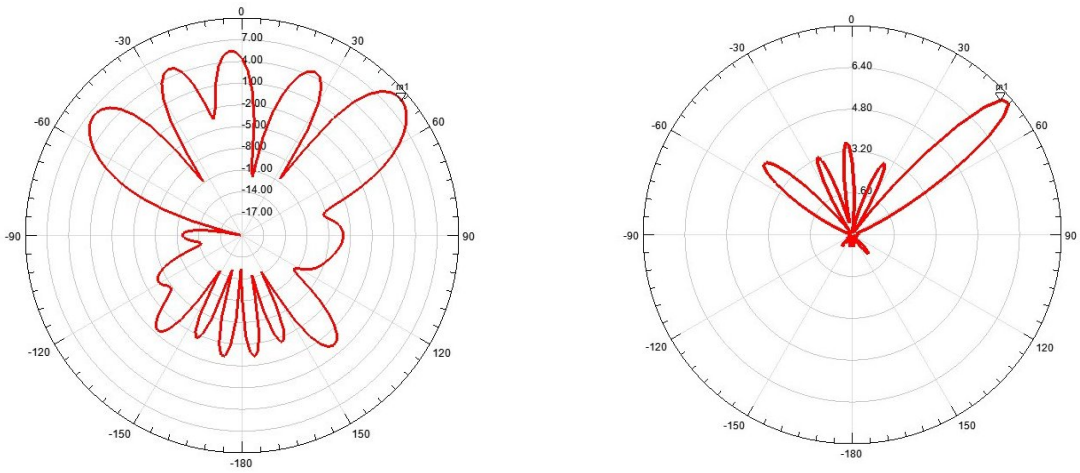
Figura 22 – Cálculo do holograma para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 45$ e $\phi = 90$. (a) Fase obtida através da retro-propagação em cada elemento da metasuperfície em graus. (b) Holograma encontrado do padrão de interferência onde em 1 (branco) são as antenas que devem irradiar e em 0 (preto) são as antenas que não devem irradiar.



(a) (b)

Fonte: Autoria própria.

Figura 23 – Diagramas de radiação simulados no HFSS para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 45$ e $\phi = 90$. (a) Diretividade em dB. (b) Diretividade em escala linear.



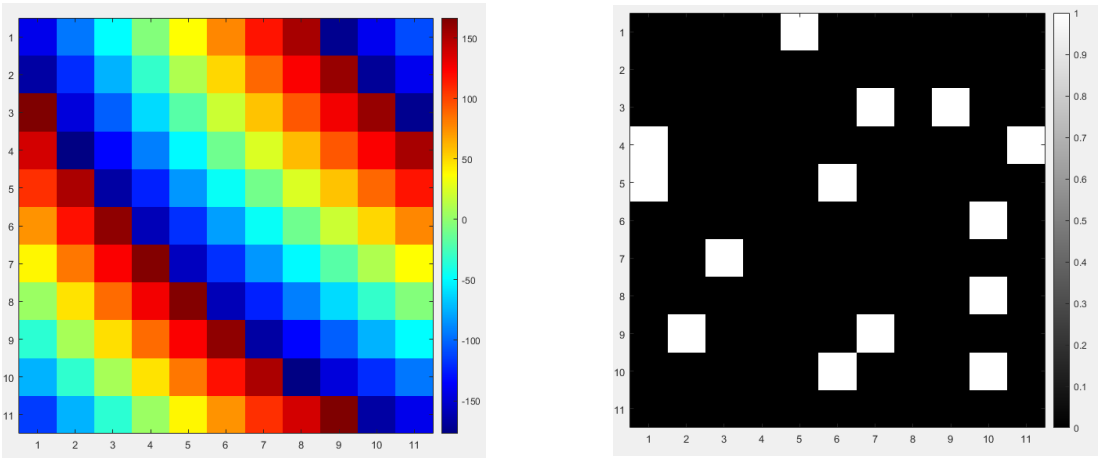
(a) (b)

Fonte: Autoria própria.

5.1.3 Apontamento de feixe,em coordenadas esféricas, para os ângulos

$\theta = 45, \phi = 45$

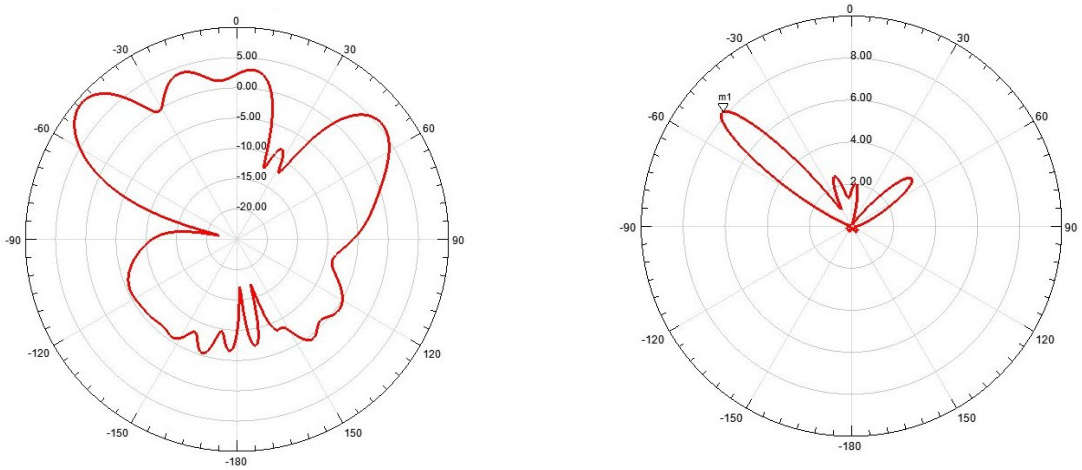
Figura 24 – Cálculo do holograma para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 45$ e $\phi = 45$. (a) Fase obtida através da retro-propagação em cada elemento da metasuperfície em graus. (b) Holograma encontrado do padrão de interferência onde em 1 (branco) são as antenas que devem irradiar e em 0 (preto) são as antenas que não devem irradiar.



(a) (b)

Fonte: Autoria própria.

Figura 25 – Diagramas de radiação simulados no HFSS para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 45$ e $\phi = 45$. (a) Diretividade em dB. (b) Diretividade em escala linear.



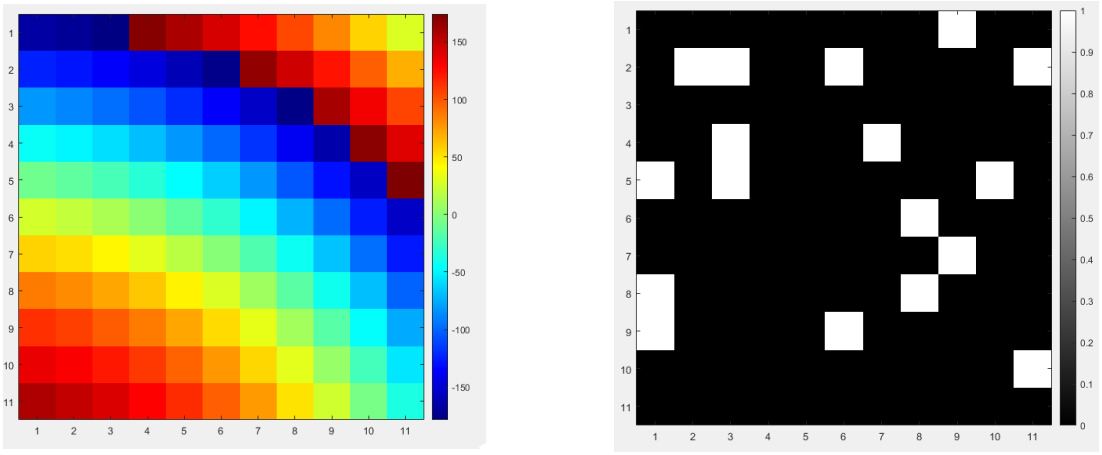
(a) (b)

Fonte: Autoria própria.

5.1.4 Apontamento de feixe,em coordenadas esféricas, para os ângulos

$\theta = 30, \phi = 120$

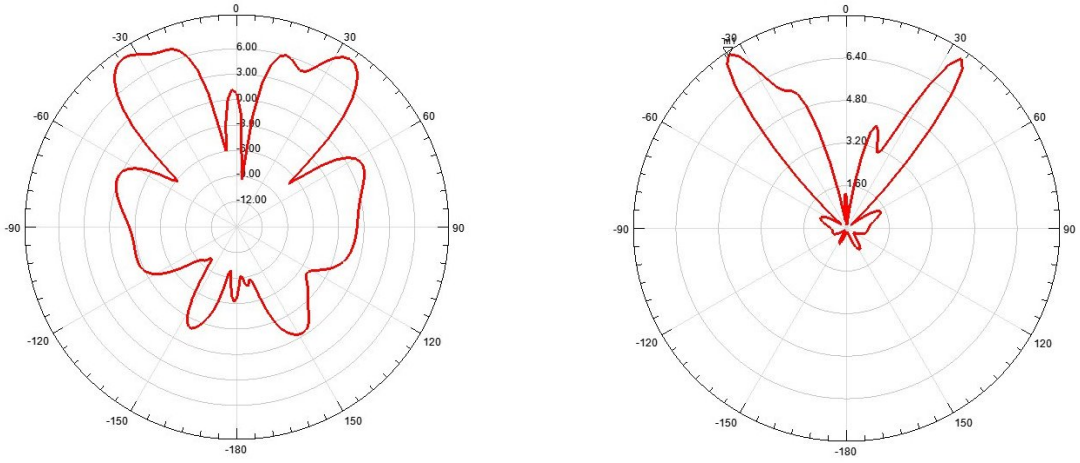
Figura 26 – Cálculo do holograma para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 30$ e $\phi = 120$. (a) Fase obtida através da retro-propagação em cada elemento da metasuperfície em graus. (b) Holograma encontrado do padrão de interferência onde em 1 (branco) são as antenas que devem irradiar e em 0 (preto) são as antenas que não devem irradiar.



(a) (b)

Fonte: Autoria própria.

Figura 27 – Diagramas de radiação simulados no HFSS para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 30$ e $\phi = 120$. (a) Diretividade em dB. (b) Diretividade em escala linear.



(a) (b)

Fonte: Autoria própria.

5.1.5 Análise dos Resultados

Através dos resultados apresentados, é visível que a antena possui o seu funcionamento confirmado. Com sua reconfiguração funcionando de maneira correta, apontando os feixes de maior potência para a direção desejada. Entretanto, os lóbulos secundários apresentados estão bastante elevados, passando de 5dB para o apontamento em algumas direções, sendo que é recomendado que estes estejam pelo menos abaixo de 0dB, pois

quanto maior os lóbulos secundários, significa que mais potência está sendo irradiada em direções indesejadas, e potência é um dos bens mais caros quando se está trabalhando com circuitos de RF. Portanto é necessário que se realize processos de otimização, de modo a conseguir diminuir estes lóbulos secundários.

Isso se deve em partes, ao fato de que se utiliza o método da retro-propagação, com a aplicação de um limite de diferença de fase para encontrar quais antenas devem ser ligadas ou não, ou seja, o holograma gerado não é igual ao padrão de interferência calculado, mas sim uma aproximação. Para gerar um holograma mais próximo do calculado, seria necessário utilizar elementos irradiantes que permitem um controle da potência irradiada por cada antena. É notado inclusive, que os feixes apontados em algumas direções, por exemplo, $\theta = 30, \phi = 120$, há ainda, além dos feixes mais fracos em direções indesejadas, um feixe na direção oposta de ϕ, isso se deve ao fato de no método da retro-propagação, alguns apontamentos diferentes de feixes possuírem fase bem semelhante, em vários elementos, ocorrendo principalmente em feixes em direções (θ, ϕ) e , $(\theta, \phi + \pi)$ como fica bem evidente nos resultados apresentados.

Outro fator que influencia no apontamento de feixe é o fato que durante a maior parte do desenvolvimento do projeto, foram escolhidos elementos de modo a se conseguir um melhor casamento de impedância (o qual não é mostrado, pois este varia para cada feixe que está sendo apontado), como por exemplo o uso de um conector coaxial lateral como alimentador ao invés de um conector centralizado na antena, o que fez com que a potência não estivesse propagada simetricamente pela antena, o que faz com que a potência irradiada entre alguns elementos varie bruscamente, interferindo no apontamento do feixe.

Tendo em vista o problema dos lóbulos secundários elevados, foi proposto o uso da CNN e o uso de técnicas de otimização convexa para se obter um holograma ótimo, para que seja possível uma melhoria no apontamento dos feixes. É importante ressaltar que as otimizações serão realizadas sobre os fatores de arranjo calculados a serem apresentados na próxima seção. Já para os problemas referentes ao design da antena, será realizado futuramente um processo de otimização nos elementos irradiantes, e conector de alimentação da antena, de modo a se atingir uma antena com uma melhor capacidade de apontamento de feixes.

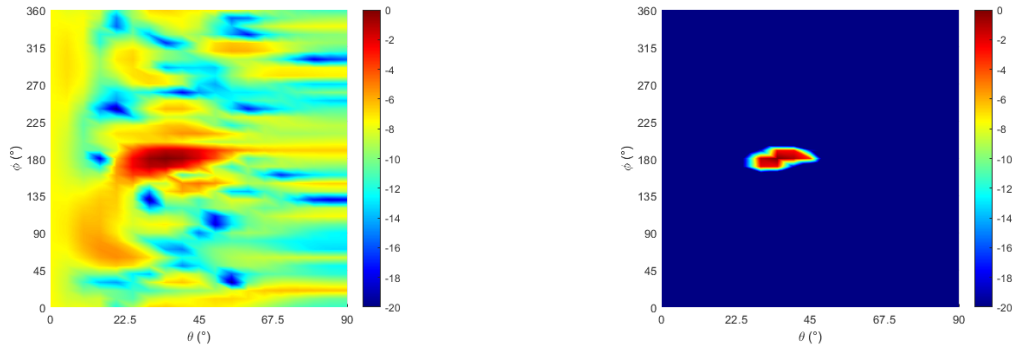
5.2 Processo de Otimização

Para o processo de otimização convexa, foi utilizada o pacote CVX de GRANT e BOYD (2020). Este pacote foi utilizado para implementar a Equação (2.24), de modo a conseguir obter o holograma ótimo. Para a aplicação desta equação, para a aplicação desta equação foram tidos $\alpha = 0.5$ que, como já explicado na Seção 2, é o valor necessário

para que a solução do problema seja binária, e $\gamma = 100$, foi obtido de maneira empírica, testando diversos valores para essa variável, e este foi o valor para o qual foram obtidos os melhores resultados em termos de redução dos lóbulos secundários para um grande diversidade de apontamentos de feixe.

Já para o processo de otimização com o uso da CNN, após o processo de treinamento com os dados obtidos pelo método da retro-propagação, é determinada a imagem de entrada desta com o fator de arranjo desejado, para que assim, a CNN consiga identificar o holograma necessário para gerar este fator de arranjo. A imagem introduzida na CNN referente ao fator de arranjo desejado é, basicamente, o fator de arranjo obtido pelo método da retro-propagação para a direção desejada, porém, com uma redução, dos lóbulos secundários presentes neste. Uma representação com imagens do fator de arranjo obtido pelo método da retro-propagação original e deste com os lóbulos secundários reduzidos está apresentado na Figura 28, na qual, os fatores de arranjo são trabalhados de maneira normalizada, de modo a conseguir trabalhar direta e unicamente sobre a redução dos lóbulos secundários, sem ter um foco em atingir ganhos específicos.

Figura 28 – Fator de Arranjo Obtido Pelo Método da Retro-Propagação. (a) Calculado. (b) Com Lóbulos Secundários Reduzidos Manualmente.



(a) (b)

Fonte: Autoria própria.

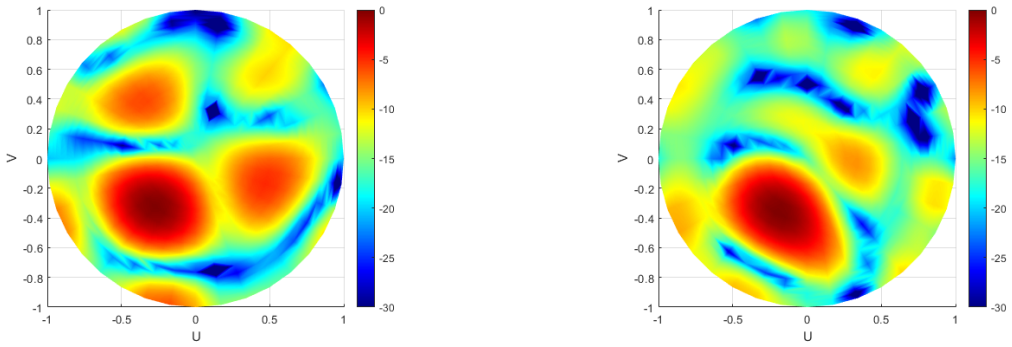
Para ser mais específico, durante o processo de redução dos lóbulos secundários das imagens dos fatores de arranjo, são mantidos os valores calculados do fator de arranjo que estão a até 4dB abaixo do valor máximo (referente ao valor máximo do lóbulo principal). Os valores que estão a até 7dB abaixo do valor máximo do fator de arranjo, são reduzidos para 20% de seu valor original, e os demais valores do fator de arranjo são reduzidos para 10% de seu valor original. Estas reduções foram realizadas de modo a manter o lóbulo principal ainda robusto, para que este possa ser identificado pela CNN, mas sem manter um alto nível de lóbulos secundários. Os valores usados para as reduções dos lóbulos secundários utilizados foram encontrados de maneira empírica, realizando testes para uma grande diversidade de apontamentos diferentes.

5.2.1 Resultados Para a Antena 11x11

Para a antena 11x11, foram realizados testes para diversos apontamentos de feixe para as duas técnicas de otimização. Entretanto, o procedimento de otimização convexa não apresentou bons resultados. Para a grande maioria dos apontamentos testados, o pacote CVX apresenta como resposta que a solução do problema é inviável, e para os apontamentos para os quais este retornou uma solução, ele retornou também que houve uma alta divergência da solução objetiva, e, quando aplicadas estas soluções para o cálculo do fator de arranjo, os resultados obtidos divergiram muito do desejado, e não serão apresentados por não agregar em nada ao trabalho.

Já os resultados referentes aos hologramas e fatores de arranjo obtidos pelo método da retro-propagação e os dados da otimização obtida através da CNN estão apresentados nas Figuras 29 - 34. Os dados estão apresentados no espaço dos senos (BROWN, 2012), que não possuem uma fácil análise em relação ao ângulo para o qual os feixes estão apontados. Porém, esta representação facilita a análise referente a como a antena está irradiando potência no espaço, facilitando a visualização do lóbulo primário e dos lóbulos secundários.

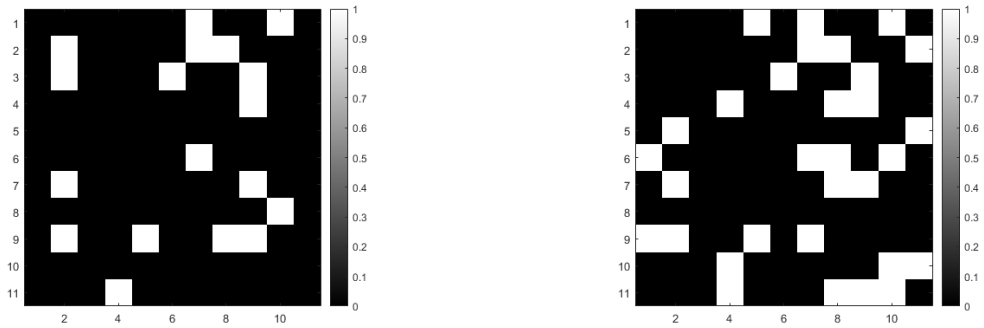
Figura 29 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 23$ e $\phi = 230$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN.



(a) (b)

Fonte: Autoria própria.

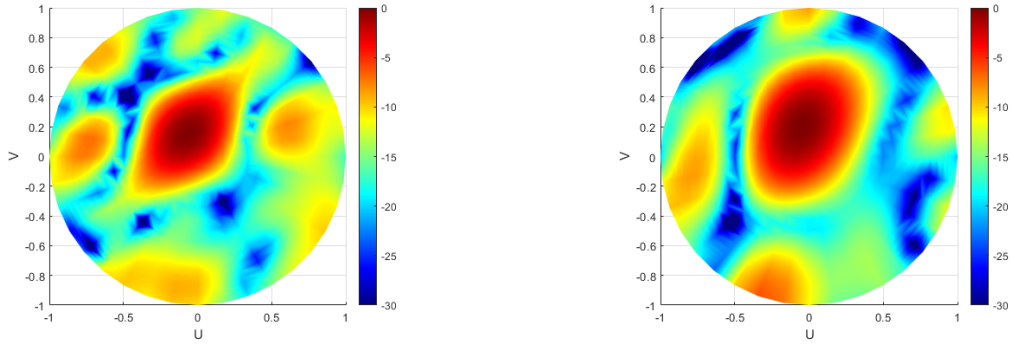
Figura 30 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 23$ e $\phi = 230$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN.



(a) (b)

Fonte: Autoria própria.

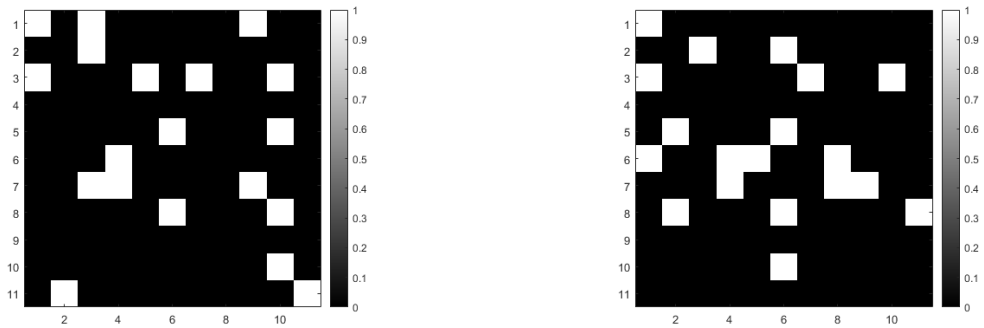
Figura 31 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 10$ e $\phi = 115$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN.



(a) (b)

Fonte: Autoria própria.

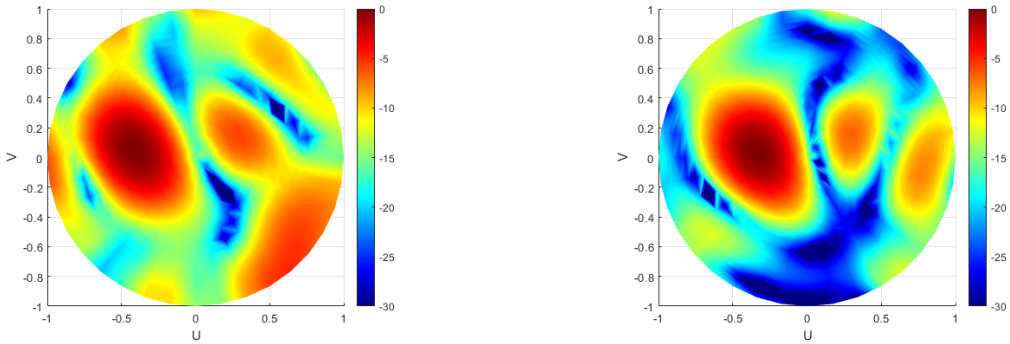
Figura 32 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 10$ e $\phi = 115$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN.



(a) (b)

Fonte: Autoria própria.

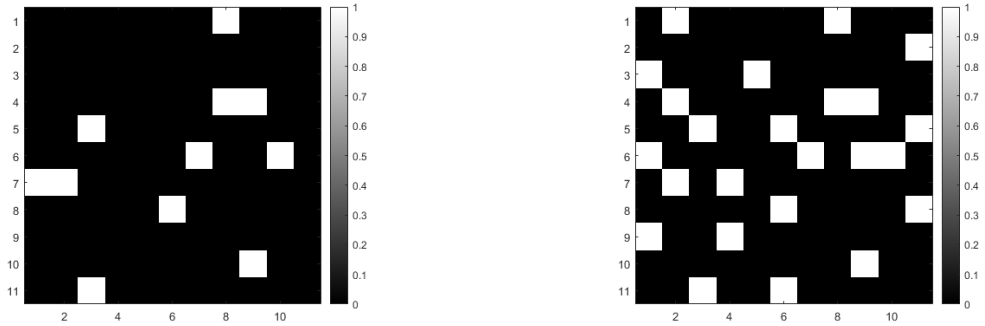
Figura 33 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 25$ e $\phi = 180$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN.



(a) (b)

Fonte: Autoria própria.

Figura 34 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 25$ e $\phi = 180$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN.



(a) (b)

Fonte: Autoria própria.

Através destes dados, é possível observar que há uma redução considerável nos lóbulos secundários do fator de arranjo para os apontamentos testados, apresentando uma otimização nos resultados, como era desejado. Porém, é possível observar, também, que apesar de haver uma redução nos lóbulos secundários, há para alguns casos, como o apresentado na Figura 31, uma modificação no lóbulo primário, no caso, um alargamento do feixe, o que é um resultado indesejado. Isto indica que o método da CNN, apesar de apresentar alguns bons resultados, no que diz respeito a redução dos lóbulos secundários, não é um método 100% confiável, quando se está trabalhando com esta antena.

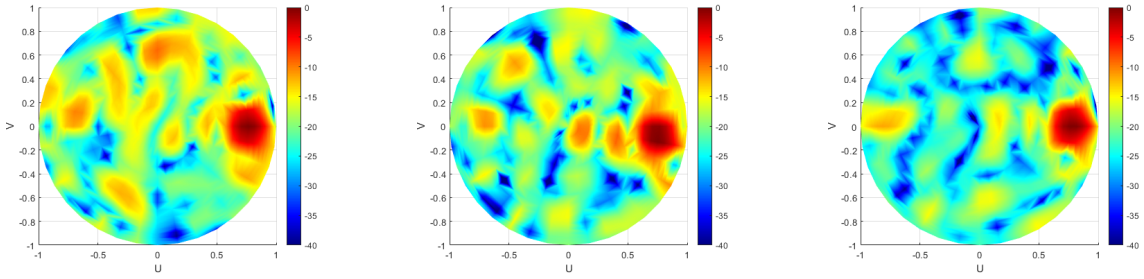
5.2.2 Resultados Para a Antena 20x20

Para a antena com 20x20 elementos irradiantes, foi possível comprovar o funcionamento do procedimento de otimização convexa, com o pacote CVX conseguindo encontrar

uma solução otimizada para todos os apontamentos de feixes testados. E, assim como para a antena com 11x11 elementos irradiantes, foi possível realizar otimizações utilizando o método da CNN.

Uma comparação para os resultados obtidos pelo método da retro-propagação e pelas duas técnicas de otimização estão apresentados nas Figuras 35 - 40.

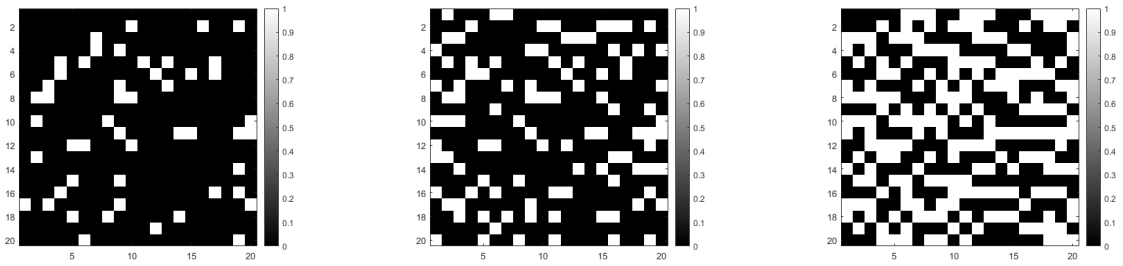
Figura 35 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 50$ e $\phi = 0$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. (c) Obtido pelo método da otimização convexa.



(a) (b) (c)

Fonte: Autoria própria.

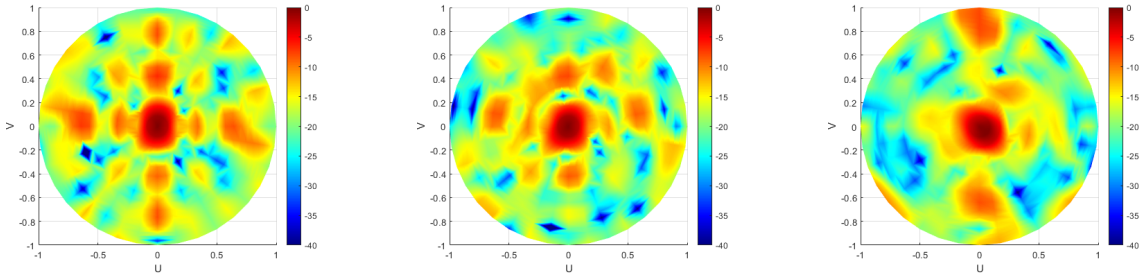
Figura 36 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 50$ e $\phi = 0$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN (c) Obtido pelo método da otimização convexa.



(a) (b) (c)

Fonte: Autoria própria.

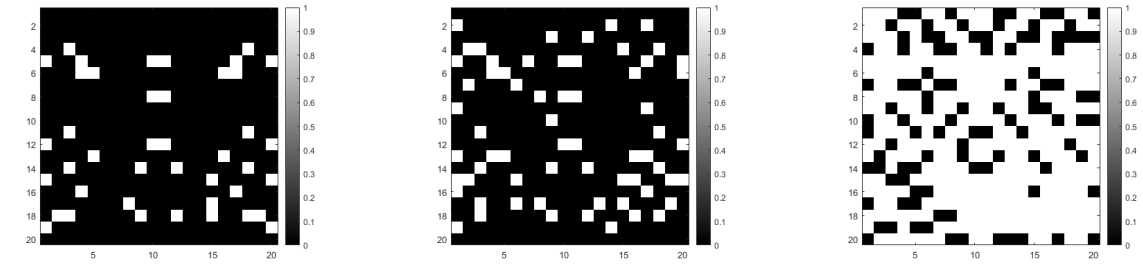
Figura 37 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 0$ e $\phi = 0$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. (c) Obtido pelo método da otimização convexa.



(a) (b) (c)

Fonte: Autoria própria.

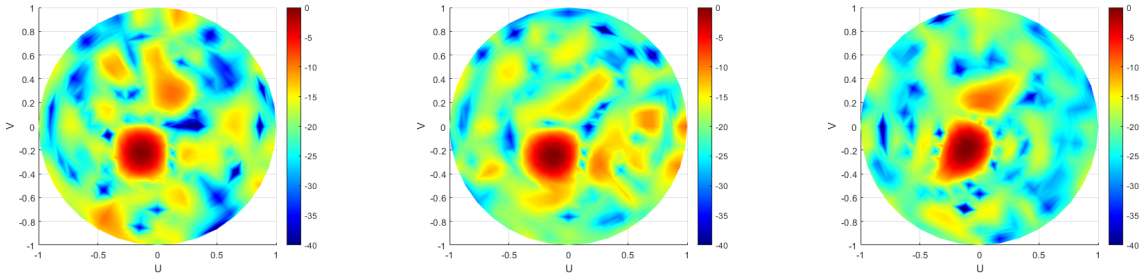
Figura 38 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 0$ e $\phi = 0$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN (c) Obtido pelo método da otimização convexa.



(a) (b) (c)

Fonte: Autoria própria.

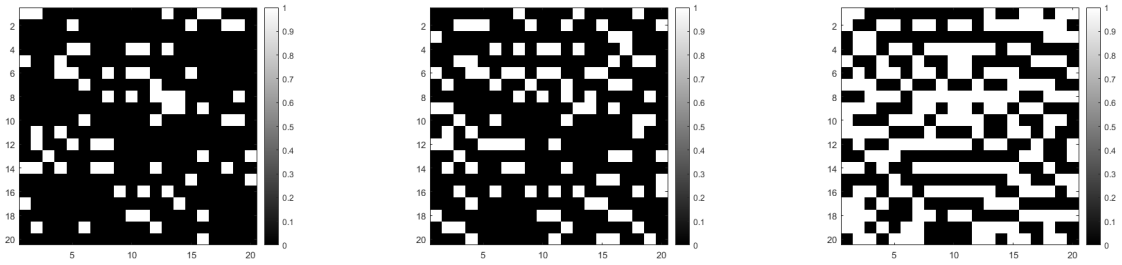
Figura 39 – Fator de arranjo obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 15$ e $\phi = 240$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN. (c) Obtido pelo método da otimização convexa.



(a) (b) (c)

Fonte: Autoria própria.

Figura 40 – Holograma obtido, para o apontamento, em coordenadas esféricas, para os ângulos $\theta = 15$ e $\phi = 240$. (a) Obtido pelo método da retro-propagação. (b) Obtido pela CNN (c) Obtido pelo método da otimização convexa.



(a) (b) (c)

Fonte: Autoria própria.

Através destes dados, é possível observar que as duas técnicas de otimização apresentam um bom funcionamento. Com a otimização convexa apresentando uma maior qualidade na redução dos lóbulos secundários, mas a CNN também apresenta bons resultados. Com isso, é identificado que para esta antena, a alternativa que apresenta uma melhor otimização no apontamento de feixe é o método da otimização convexa.

Se comparados os dados obtidos para as duas antenas, é observado que para a antena com mais elementos, se tem uma maior precisão no apontamento do feixe, com um lóbulo principal bastante estreito, se comparado com a antena com uma menor quantidade de elementos. Já em se tratando dos métodos de otimização, é observado que o uso de CNN é viável para ambos os casos, se mostrando mais versátil. Já o método da otimização convexa, se aproveita de uma antena com uma maior quantidade de elementos, pois tem uma maior liberdade para otimização.

6 Conclusão

O uso de técnicas de holografia, juntamente com metasuperfícies, embora seja uma área que não foi explorada o suficiente a ponto de possuir uma quantidade de aplicações na indústria tão grande quanto outros métodos, trata-se de uma técnica bastante promissora que promete trazer grandes inovações para a área de circuitos de alta frequência.

Para o desenvolvimento da antena apresentada neste projeto, foi realizada inicialmente escolhas para alimentadores, substrato como meio propagante, elementos irradiantes e elementos chaveadores. Sendo estes projetados separadamente, e por fim implementados em um circuito completo, faltando serem implementados ainda, os circuitos de polarização dos diodos PIN.

Após a implementação do circuito completo, foram realizados testes de apontamento de feixes para a antena desenvolvida, utilizando o método da retro-propagação. O circuito final, conseguiu mostrar bom funcionamento da sua reconfigurabilidade, conseguindo apontar feixes em diversas direções. Entretanto, não apresentou bons resultados no que diz respeito a qualidade de apontamento de feixes, devido ao alto nível dos lóbulos secundários. Para uma otimização neste apontamento, foi desenvolvida uma rede neural convolucional para ser aplicada ao método de apontamento, e utilizadas, ainda, técnicas de otimização convexa para uma melhoria neste apontamento de feixe.

A rede neural convolucional foi desenvolvida para realizar uma otimização sobre o fator de arranjo da antena, e consequentemente otimizar o padrão de radiação da antena. A CNN recebe, no seu treinamento, como entradas o fator de arranjo obtido do holograma calculado pelo método da retro-propagação, e como rótulos os hologramas necessários para gerar este fator de arranjo. E, durante sua etapa de classificação, é aplicado em sua entrada o fator de arranjo desejado, obtido ao reduzir os lóbulos secundários do fator de arranjo obtido pelo método da retro-propagação, de modo que a CNN consiga gerar em sua saída os dados referentes ao holograma que melhor se adéque para a geração deste fator de arranjo otimizado.

A otimização do apontamento do fator de arranjo foi observado como um problema de otimização convexa. Isto permitiu que fosse utilizada esta técnica de otimização para se obter uma redução nos lóbulos secundários observados durante a solução utilizando o método da retro-propagação.

Ambas as técnicas de otimização apresentam bons resultados. Entretanto, para uma antena com 11x11 elementos irradiantes, o solucionador utilizado para a otimização convexa apresentou que sua solução era inviável, e portanto, foi utilizada apenas a otimização pelo método da CNN. Já para uma antena com 20x20 elementos irradiantes, a

técnica de otimização convexa apresentou resultados bastante satisfatórios, melhores que os resultados obtidos pela otimização com o uso da CNN, demonstrando ser assim, uma técnica com maior qualidade, enquanto a CNN apresentou uma maior flexibilidade.

6.1 Trabalhos Futuros

Estando faltando para a conclusão do projeto, o desenvolvimento de algumas etapas, que devem ser realizadas em uma certa ordem para garantir uma organização nesta etapa de final do projeto.

Primeiramente, levando em consideração os resultados não tão bons para apontamento de determinados feixes, deve ser realizada uma otimização nas antenas. Com o objetivo de se obter uma redução dos lóbulos secundários observados nos diagramas de radiação. Caso a antena continue com estes problemas, se torna inviável sua aplicação em um sistema de comunicação, por exemplo, pois isso resultaria em uma grande perda de dados durante seu funcionamento, o que é sempre indesejado.

Devem ser, então, desenvolvidos os circuitos de polarização para os diodos PIN responsáveis por controlar o os elementos que devem ou não irradiar potência. Pois caso estes circuitos não sejam desenvolvidos, os diodos PIN não funcionam corretamente como chaveadores.

Com o objetivo de se obter uma maior especificação no que diz respeito à comparação entre as técnicas de otimização, será realizada uma análise de complexidade de ambas as técnicas, de modo a identificar como o tempo necessário para solucionar o problema evolui a medida que o número de elementos irradiantes aumenta. Pois, a depender de como isto ocorre, é possível que a CNN se torne uma melhor alternativa que a otimização convexa para aplicações onde é priorizada a velocidade no apontamento do feixe ao invés da qualidade no apontamento.

Referências

AMBS, P.; HUIGNARD, J.; LOISEAUX, B. *Encyclopedia of Condensed Matter Physics*. Academic Press: [s.n.], 2005. v. 1. 332-341 p. Citado na página 29.

BALANIS, C. A. *Antenna Theory*. Wiley, Nova Jersey: [s.n.], 2005. v. 3. Citado 4 vezes nas páginas 21, 22, 23 e 25.

BHATTACHARYYA, A. K. *Phased Array Antennas - Floquet Analysis, Synthesis, BFNs, and Active Array Systems*. Wiley: [s.n.], 2006. v. 1. Citado 2 vezes nas páginas 23 e 39.

BISWAL, A. Convolutional neural network tutorial. Simplilearn, 2021. Citado 3 vezes nas páginas 35, 36 e 37.

BLACK, E. J. Holographic beam forming and MIMO. Pivotal, 2017. Citado 2 vezes nas páginas 19 e 24.

BOYD, S.; VANDENBERGHE, L. *Convex Optimization*. Cambridge University Press: [s.n.], 2004. v. 1. Citado na página 39.

BROWN, A. D. *Eletronically Scanned Arrays - MATLAB Modeling and Simulation*. CRC-Press: [s.n.], 2012. v. 1. Citado na página 62.

FAENZI, M. et al. Metasurface antennas: New models, applications and realizations. Nature Scientific Reports, 2019. Citado 3 vezes nas páginas 18, 32 e 42.

FONG, B. H. et al. Scalar and tensor holographic artificial impedance surfaces. IEEE Transactions on Antennas and Propagation, v. 58, p. 3212–3221, 2010. Citado 3 vezes nas páginas 18, 32 e 33.

FUCHS, B.; RONDINEAU, S. Array pattern synthesis with excitation control via norm minimization. IEEE Transactions on Antennas and Propagatios, 2016. Citado 3 vezes nas páginas 19, 39 e 40.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, Massachusetts: [s.n.], 2016. v. 1. Citado 5 vezes nas páginas 34, 35, 36, 37 e 38.

GRANT, M.; BOYD, S. Cvx: Matlab software for disciplined convex programming - version 2.2, build 1148. 2020. Disponível em: <http://cvxr.com/cvx/>. Citado na página 60.

HARIHARAN, P. *Basics of Interferometry*. Elsevier, Sydney: [s.n.], 2007. v. 2. 212 p. Citado 2 vezes nas páginas 30 e 31.

KEYSIGHT TECHNOLOGIES. *Application note*: Understanding RF/microwave solid state switches and their applications. [S.l.], 2017. Citado 5 vezes nas páginas 19, 27, 28, 44 e 45.

KHAN, S. et al. *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool Publishers: [s.n.], 2018. v. 1. Citado 2 vezes nas páginas 37 e 38.

LIZUKA, K. *Engineering Optics*. Springer, Toronto: [s.n.], 2019. v. 4. 203-239 p. Citado na página 29.

MAILLOUX, R. J. *Phased Array Antenna Handbook*. Artech House: [s.n.], 2018. v. 3. Citado 2 vezes nas páginas 22 e 23.

MENCAGLI, J. M.; MARTINI, E.; MACI, S. Surface wave dispersion for anisotropic metasurfaces constituted by elliptical patches. IEEE Transactions on Antennas and Propagatios, 2015. Citado na página 32.

MOHANTA, H. C.; KOUZANI, A. Z.; MANDAL, S. K. Reconfigurable antennas and their applications. Universal Journal of Electrical and Electronic Engineering, v. 6, p. 239–258, 2010. Citado 3 vezes nas páginas 18, 19 e 24.

OVEJERO, D. G.; MACI, S. Gaussian ring basis functions for the analysis of modulated metasurface antennas. IEEE Transactions on Antennas and Propagation, 2015. Citado na página 18.

POZAR, D. M. *Microwave Engineering*. Wiley: [s.n.], 2012. v. 4. Citado 2 vezes nas páginas 27 e 28.

ROGERS CORPORATION. *Kappa 438 Laminates*: Data sheet. Chandler - Arizona, 2019. Citado na página 41.

RUSCH, C. *Handbook of Antenna Technologies*. Springer, Singapura: [s.n.], 2016. v. 1. 2689-2725 p. Citado 2 vezes nas páginas 29 e 30.

SANTANA, M. P. *Reconfigurable Metasurface Antenna Array Using Holographic Beamforming*. 106 p. Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2021. Citado 2 vezes nas páginas 19 e 40.

SHANG, G. et al. Metasurface holography in the microwave regime. Photonics, v. 8, 135, p. 18, 2021. Citado na página 42.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. ICLR, 2015. Citado 2 vezes nas páginas 18 e 52.

STUTZMAN, W. L.; THIELE, G. A. *Antenna Theory and Design*. Wiley, West Virginia: [s.n.], 2013. v. 3. Citado na página 22.

TAO, S. et al. Coding programmable metasurfaces based on deep learning techniques. IEEE Journal, v. 10, p. 114–125, 2020. Citado 3 vezes nas páginas 18, 42 e 52.

VERMA, Y. A complete understanding of dense layers in neural networks. 2021. Disponível em: <https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/>. Citado na página 38.

YANG, H. et al. A programmable metasurface with dynamic polarization, scattering and focusing control. Nature Scientific Reports, 2016. Citado na página 42.

YURDUSEVEN, O. et al. Design and analysis of a reconfigurable holographic metasurface aperture for dynamic focusing in the fresnel zone. IEEE Access, v. 5, p. 15055–15065, 2017. Citado 7 vezes nas páginas 18, 24, 25, 26, 33, 42 e 48.

Apêndices

APÊNDICE A – Códigos Desenvolvidos Para o Cálculo do Holograma da Antena Pelo Método da Retro Propagação

```
clear all;
close all;
clc;

%%
% Definindo variáveis

PIXELS = 20;
N = PIXELS; %Número de elementos em x.
M = PIXELS; %Número de elementos em y.
f = 12e9; % Freq de operação.
c = 3e8; %Velocidade da luz.
lambda = c/f; % Comprimento de onda.
er = 4.5;%constante dielétrica.
lambda_g = lambda/sqrt((er+1)/2); % Comprimento de onda guiado.
dx = lambda_g/3; %Espaçamento entre os elementos no eixo x
dy = lambda_g/3; %Espaçamento entre os elementos no eixo y.
%%
% Cálculo do Holograma usando a sintese de arranjo

phase_threshold = 25; % Limiar do módulo dasfases a serem consideradas.

%%
% Descrição da Coordenadas
load('C:/Users/wemer/Desktop/tcc/Arquivos/Testes_Feixes/opt_test_bp.mat')
load('C:/Users/wemer/Desktop/tcc/Arquivos/Testes_Feixes/opt_test_cvx.mat')
Holo_cnn_bp = holo_bp;

%Holo_cnn_cvx = [1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,0,0,0,0,0;↙
1,0,1,0,0,0,1,1,1,0,1,0,1,0,1,1,1,1,1,1,1;0,1,1,1,1,1,0,0,0,0,1,0,1,0,1,1,1,0,1,1,0;↙
0,1,0,1,0,1,0,1,0,0,0,1,1,1,0,1,0,0,0,1;1,1,1,0,1,0,1,1,1,1,1,0,1,1,1,0,1,1,1,1;↙
1,0,1,0,1,0,1,1,1,0,1,0,1,0,0,1,1,0,0,0;0,0,1,1,0,1,0,1,0,1,0,1,1,1,1,1,0,1,1,1,1;↙
1,1,1,1,1,1,1,0,0,1,1,1,0,0,0,1,0,1,1,0;1,0,1,0,0,0,1,1,1,0,1,0,1,0,1,1,1,1,1,1,1;↙
0,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,0,1,1,0,1;0,0,0,1,0,1,1,1,0,0,0,1,1,1,0,1,0,0,1,0;↙
1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,1,1;1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,0,0,1,0,1,0,0;↙
0,0,0,1,0,1,0,1,1,0,0,0,1,1,1,1,1,0,0,1,1;1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1,0,1;↙
1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0;0,1,0,0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,0,0,0;↙
0,0,0,1,1,1,1,1,1,1,1,0,0,1,0,1,1,0,0,1,1;1,1,1,1,1,1,0,0,1,1,1,1,1,0,1,1,0,1,1,1,1;↙
1,0,1,0,1,0,1,1,0,1,0,1,1,0,1,0,1,1,1,1];

Holo_cnn_cvx = holo_cvx;
theta_p = 15; %Elevação do ponto a ser projetado.
phi_p = 240; % Azimute do ponto a ser projetado.


theta0 = 0:pi/36:pi; % Elavação
phi0 = 0:pi/18:2*pi; % Azimute
[phi,theta] = meshgrid(phi0,theta0); % Malha nas coordenadas.
sinU = sin(theta).*cos(phi); % U-sine space.
sinV = sin(theta).*sin(phi); % V-sine space.

%%
% Posicionamento das Antenas

Px = -(N-1)/2:1:(N-1)/2; % Posição de cada antena no eixo x.
```

```
Py = -(-(M-1)/2:1:(M-1)/2); % Posição de cada antena no eixo y.
Px3d = dx*meshgrid(Px); % Matriz de posições.
Py3d = dy*meshgrid(Py)'; % Matriz de posições.


%%
%leitura h

H_HFSS = table2array(readtable('H20.csv')); % Importa os dados da antena simualada.


H_HFSS(isnan(H_HFSS)) = 0;


Hy_HFSS = rot90((reshape(H_HFSS(:,4),N,M)-1i*reshape(H_HFSS(:,5),N,M))');
Hx_HFSS = rot90((reshape(H_HFSS(:,6),N,M)-1i*reshape(H_HFSS(:,7),N,M))');
Hz_HFSS = rot90((reshape(H_HFSS(:,8),N,M)-1i*reshape(H_HFSS(:,9),N,M))');
magH = abs(sqrt(Hx_HFSS.^2+Hy_HFSS.^2+Hz_HFSS.^2));


%%
% Projeção do beam no arranjo.

rho = 1; % Distância radial do ponto a ser projetado

x0 = rho*sind(theta_p)*cosd(phi_p); % Coordenada X do ponto a ser projetado.
y0 = rho*sind(theta_p)*sind(phi_p); % Coordenada Y do ponto a ser projetado.
z0 = rho*cosd(theta_p); % Coordenada Z do ponto a ser projetado.

r_linha = [x0 y0 z0]; % Coordenadas do ponto a ser projetado.
beam_projection = zeros(N,M); % Criação do array da projeção.

for n = 1:N
    for m = 1:M
        % Cálculo da projeção sobre o arranjo usando onda esférica.
        beam_projection(n,m) = exp(1i*(2*pi/lambda)*norm([Px3d(n,m) Py3d(n,m) 0]-↙
r_linha))/norm([Px3d(n,m) Py3d(n,m) 0]-r_linha);
    end
end

beam_phase = angle(beam_projection)*180/pi;% Cálculo da fase da projeção sobre o↙
arranjo.

%%
% Cálculo do holograma usando a projeção no arranjo

 Hologram_phase_2 = beam_phase-180*angle(Hy_HFSS)/pi; % Gravação da fase do holograma.

 Holo2 = zeros(N,M); % Criação do array do holograma usando projeção.
 n_ant_2 = 0;
   for n = 1:1:N
       for m = 1:1:M
           if abs(Hologram_phase_2(n,m)) <= phase_threshold  % Verificação se esta↙
dentor do limiar.
                Holo2(n,m) = 1; % Ligando a antena no holograma.
```

```
                n_ant_2 = n_ant_2+1;
            else
                continue;
            end
        end
    end
Holo2(ceil(N/2),ceil(M/2)) = 0;

%%

Sreal2 = (Holo2).*exp(  1i*angle(Hy_HFSS)); %Matriz de excitação usando a fasse da cavidade e a máscara do holograma.

afr2 = zeros(length(theta0),length(phi0)); % Inicio do vetor do AFr.
for n = 1:1:N
    for m =1:1:M
        afr2 =afr2+Sreal2(n,m).*exp((1i*2*pi/lambda)*(Px3d(n,m)*sinU+Py3d(n,m)*sinV));% Cálculo do fator de arranjo.
    end
end

APr2 = afr2; % cálculo do diagrama de radiação do arranjo usando a projeção de onda esférica.

%%
%teste resultado cnn

%Holo_cnn_bp = zeros(N,M);%new
Sreal_cnn_bp = (Holo_cnn_bp).*exp(  1i*angle(Hy_HFSS)); %Matriz de excitação usando a fasse da cavidade e a máscara do holograma.

afr_cnn_bp = zeros(length(theta0),length(phi0)); % Inicio do vetor do AFr.
for n = 1:1:N
    for m =1:1:M
        afr_cnn_bp =afr_cnn_bp+Sreal_cnn_bp(n,m).*exp((1i*2*pi/lambda)*(Px3d(n,m)*sinU+Py3d(n,m)*sinV));% Cálculo do fator de arranjo.
    end
end

APr_cnn_bp = afr_cnn_bp; % cálculo do diagrama de radiação do arranjo usando a projeção de onda esférica.

%%
%teste resultado cnn

%Holo_cnn_bp = zeros(N,M);%new
Sreal_cnn_cvx = (Holo_cnn_cvx).*exp(  1i*angle(Hy_HFSS)); %Matriz de excitação usando a fasse da cavidade e a máscara do holograma.

afr_cnn_cvx = zeros(length(theta0),length(phi0)); % Inicio do vetor do AFr.
for n = 1:1:N
    for m =1:1:M
        afr_cnn_cvx =afr_cnn_cvx+Sreal_cnn_cvx(n,m).*exp((1i*2*pi/lambda)*(Px3d(n,m)*sinU+Py3d(n,m)*sinV));% Cálculo do fator de arranjo.
    end
```

```
end

APr_cnn_cvx = afr_cnn_cvx; % cálculo do diagrama de radiação do arranjo usando a projeção de onda esférica.
%%
%Feixe "ótimo"
feixe_principal = (20*log10(abs(afr2)./max(max(abs(afr2))))>-7);
opt_test = (20*log10(abs(afr2)./max(max(abs(afr2))))>-4).*(abs(afr2))*0.8+ ...
           (20*log10(abs(afr2)./max(max(abs(afr2))))>-7).*(abs(afr2))*0.2+ ...
           (20*log10(abs(afr2)./max(max(abs(afr2))))<=-7).*(abs(afr2))*0.1;



%%
%Otimização Convexa

limite_db = max(max(10*log10(abs(afr2))));
limite = 10^(limite_db/10);

theta_x = theta_p;
phi_x = phi_p;
gamma = 5;
delta = 30;

theta_0 = (pi/180)*linspace(theta_x-gamma,theta_x+gamma,5);

phi_0 = (pi/180)*linspace(phi_x-gamma,phi_x+gamma,5);

[phi_grid_0,theta_grid_0] = meshgrid(phi_0,theta_0); % Malha nas coordenadas.
sinU_0 = sin(theta_grid_0).*cos(phi_grid_0); % U-sine space.
sinV_0 = sin(theta_grid_0).*sin(phi_grid_0); % V-sine space.

%%
% Cálculo do fator de arranjo

U_0 = reshape(kron(Px3d(:),sinU_0)',length(theta_0),length(phi_0),length(Px3d(:))); % Calcula X*sinU e dá reshape em um array 3d.
V_0 = reshape(kron(Py3d(:),sinV_0)',length(theta_0),length(phi_0),length(Py3d(:))); % Calcula Y*sinV e dá reshape em um array 3d.
exp_0 = exp((1i*2*pi/lambda)*(U_0+V_0));

Hy3d = reshape(Hy_HFSS(:),1,1,length(Hy_HFSS(:))); % Dá um reshape nos pesos para alocalo na terceira dimensão.

a_0 = exp(1i*angle(Hy3d)).*exp_0; % Cálcula o diagrama.



cvx_begin
    variable  x(N,M) nonnegative
    xcvx = Holo2+abs(Holo2-1).*x;
    %xcvx = x;
    %minimize (norm(xcvx,1)+1000*M*N*norm(xcvx-0.5,inf))
    minimize 100*norm(xcvx-0.5,inf)
```

```
    subject to
       abs(sum(a_0.*repmat(reshape(xcvx(:),1,1,length(xcvx(:))),length(theta_0),length
(phi_0)),3)-limite) <= 4
cvx_end

a = abs(xcvx);
b = (a>0.7);
Holo_cvx = (abs(round(xcvx))>0.5);

Sreal_cvx = (Holo_cvx).*exp(1i*angle(Hy_HFSS)); %Matriz de excitação usando a fasse da
cavidade e a máscara do holograma.

afr_cvx = zeros(length(theta0),length(phi0)); % Inicio do vetor do AFr.
for n = 1:1:N
    for m =1:1:M
        afr_cvx =afr_cvx+Sreal_cvx(n,m).*exp((1i*2*pi/lambda)*(Px3d(n,m)*sinU+Py3d(n,
m)*sinV));% Cálculo do fator de arranjo.
    end
end


%%
%normalizando fatores de arranjo
opt_test = abs(opt_test)./(max(max(abs(opt_test))));
afr2 = abs(afr2)./(max(max(abs(afr2))));
afr_cnn_bp = abs(afr_cnn_bp)./(max(max(abs(afr_cnn_bp))));
afr_cnn_cvx = abs(afr_cnn_cvx)./(max(max(abs(afr_cnn_cvx))));
afr_cvx = abs(afr_cvx)./(max(max(abs(afr_cvx))));

Diff_bp = abs(afr2 - opt_test);
Diff_cnn_bp = abs(afr_cnn_bp - opt_test);
Diff_cnn_cvx = abs(afr_cnn_cvx - opt_test);
Diff_cvx = abs(afr_cvx - opt_test);

Diff_bp = sum(sum(Diff_bp))/37^2
Diff_cnn_bp = sum(sum(Diff_cnn_bp))/37^2
Diff_cnn_cvx = sum(sum(Diff_cnn_cvx))/37^2
Diff_cvx = sum(sum(Diff_cvx))/37^2

%%
%plots
fig_num = 1;
figure(fig_num);fig_num=fig_num+1;

 subplot(2,3,1)
surf(180*theta/pi,180*phi/pi,abs(afr2)./max(max(abs(afr2))));
colormap(jet(256));
colorbar;
caxis([0 1]);
    shading interp
    view(2);
    xlabel('\theta (°)');
    ylabel('\phi (°)');
    title('Reconstrução pela retropropagação');
    xlim([0 90]);
```

```
    xticks(0:22.5:90);
    ylim([0 360]);
    yticks(0:45:360);
    daspect([1 4 1])

  subplot(2,3,2)
%figure(fig_num);fig_num=fig_num+1;
surf(180*theta/pi,180*phi/pi,abs(afr_cnn_bp)./max(max(abs(afr_cnn_bp))));
colormap(jet(256));
colorbar;
caxis([0 1]);
    shading interp
    view(2);
    xlabel('\theta (°)');
    ylabel('\phi (°)');
    title('Reconstrução do resultado da cnn (BP)');
    xlim([0 90]);
    xticks(0:22.5:90);
    ylim([0 360]);
    yticks(0:45:360);
    daspect([1 4 1])

 subplot(2,3,3)
%figure(fig_num);fig_num=fig_num+1;
surf(180*theta/pi,180*phi/pi,abs(afr_cnn_cvx)./max(max(abs(afr_cnn_cvx))));
colormap(jet(256));
colorbar;
caxis([0 1]);
    shading interp
    view(2);
    xlabel('\theta (°)');
    ylabel('\phi (°)');
    title('Reconstrução do resultado da cnn (CVX)');
    xlim([0 90]);
    xticks(0:22.5:90);
    ylim([0 360]);
    yticks(0:45:360);
    daspect([1 4 1])

     subplot(2,3,4)
%
%figure(fig_num);fig_num=fig_num+1;
surf(180*theta/pi,180*phi/pi,abs(afr_cvx)./max(max(abs(afr_cvx))));
colormap(jet(256));
colorbar;
caxis([0 1]);
    shading interp
    view(2);
    xlabel('\theta (°)');
    ylabel('\phi (°)');
    title('Reconstrução do resultado da CVX');
    xlim([0 90]);
    xticks(0:22.5:90);
    ylim([0 360]);
    yticks(0:45:360);
```

```
    daspect([1 4 1])

    subplot(2,3,5)
    surf(180*theta/pi,180*phi/pi,abs(opt_test)./max(max(abs(opt_test))));
colormap(jet(256));
colorbar;
caxis([0 1]);
    shading interp
    view(2);
    xlabel('\theta (°)');
    ylabel('\phi (°)');
    title('Feixe Desejado');
    xlim([0 90]);
    xticks(0:22.5:90);
    ylim([0 360]);
    yticks(0:45:360);
    daspect([1 4 1])

%%
%sine space

figure(fig_num);fig_num=fig_num+1;

       subplot(2,3,1)
        surf(sinU,sinV,20*log10(abs(afr2)./max(max(abs(afr2)))));
        colormap(jet)
          caxis([-40, 0]);
        colorbar;
        view(2);
        xlabel('Sin(U)');
        ylabel('Sin(V)');
        title('Reconstrução pela retropropagação');
        shading interp;
         axis([-1 1 -1 1]);
        daspect([1 1 1])

     subplot(2,3,2)

        surf(sinU,sinV,20*log10(abs(afr_cnn_bp)./max(max(abs(afr_cnn_bp)))));
        colormap(jet)
        caxis([-40, 0]);
        colorbar;
        view(2);
        xlabel('Sin(U)');
        ylabel('Sin(V)');
        title('Reconstrução do holograma da CNN (BP)');
        shading interp;
         axis([-1 1 -1 1]);
        daspect([1 1 1])

     subplot(2,3,3)

        surf(sinU,sinV,20*log10(abs(afr_cnn_cvx)./max(max(abs(afr_cnn_cvx)))));
        colormap(jet)
        caxis([-40, 0]);
```

```
        colorbar;
        view(2);
        xlabel('Sin(U)');
        ylabel('Sin(V)');
        title('Reconstrução do holograma da CNN (CVX)');
        shading interp;
        daspect([1 1 1])

     subplot(2,3,5)

        surf(sinU,sinV,20*log10(abs(opt_test)./max(max(abs(opt_test)))));
        colormap(jet)
        caxis([-40, 0]);
        colorbar;
        view(2);
        xlabel('Sin(U)');
        ylabel('Sin(V)');
        title('Feixe Desejado');
        shading interp;
         axis([-1 1 -1 1]);
        daspect([1 1 1])

    subplot(2,3,4)

        surf(sinU,sinV,20*log10(abs(afr_cvx)./max(max(abs(afr_cvx)))));
        colormap(jet)
        caxis([-40, 0]);
        colorbar;
        view(2);
        xlabel('Sin(U)');
        ylabel('Sin(V)');
        title('Reconstrução do Holograma da Otimização Convexa');
        shading interp;
         axis([-1 1 -1 1]);
        daspect([1 1 1])


    ax = axes;
    globaltitle = title({'Fator de arranjo no espaço dos senos';''});
    ax.Visible = 'off'; % set(a,'Visible','off');
    globaltitle.Visible = 'on'; % set(t1,'Visible','on');

 %%
 %export
 opt_test = opt_test/max(max(opt_test));
 %opt_test = abs(afr2)/max(max(abs(afr2)));

 %teste
 %opt_test = (opt_test==0)*0.1+opt_test;

 thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/opt_test.mat');
 save(thisfilename,'opt_test');
 csvwrite('45.csv',Holo2);

figure;
```

```
colormap(gray(255));
subplot(2,2,1)
image(Holo2, 'CDataMapping','scaled');
colorbar;
axis equal tight;
title('Holograma Retro-Propagação');

subplot(2,2,2)
image(Holo_cnn_bp, 'CDataMapping','scaled');
axis equal tight;
colorbar;
title('Holograma CNN (BP)');

subplot(2,2,3)
image(Holo_cnn_cvx, 'CDataMapping','scaled');
axis equal tight;
colorbar;
title('Holograma CNN (CVX)');

subplot(2,2,4)
image(Holo_cvx, 'CDataMapping','scaled');
axis equal tight;
colorbar;
title('Holograma CVX');

ax = axes;
  globaltitle = title({'Hologramas obtidos pelos diferentes métodos de↙
apontamento';''});
  ax.Visible = 'off'; % set(a,'Visible','off');

globaltitle.Visible = 'on'; % set(t1,'Visible','on');
```

```
%%
%imagens tcc
  %%
 %Hologramas

 figure;

colormap(gray(255));
 image(Holo_cvx, 'CDataMapping','scaled');
axis equal tight;
colorbar;
```

```
 figure;
```

```
colormap(gray(255));
 image(Holo_cnn_bp, 'CDataMapping','scaled');
axis equal tight;
colorbar;


figure;
colormap(gray(255));

image(Holo2, 'CDataMapping','scaled');
colorbar;
axis equal tight;

%%

figure;
surf(sinU,sinV,20*log10(abs(afr_cvx)./max(max(abs(afr_cvx)))));
        colormap(jet(256))
          caxis([-40, 0]);
        colorbar;
        view(2);
        xlabel('U');
        ylabel('V');
        %title('Reconstrução do feixe "objetivo"');
        shading interp;
         axis([-1 1 -1 1]);
        daspect([1 1 1])


figure;
surf(sinU,sinV,20*log10(abs(afr_cnn_bp)./max(max(abs(afr_cnn_bp)))));
        colormap(jet(256))
          caxis([-40, 0]);
        colorbar;
        view(2);
        xlabel('U');
        ylabel('V');
        %title('Reconstrução do feixe "objetivo"');
        shading interp;
         axis([-1 1 -1 1]);
        daspect([1 1 1])

figure;
surf(sinU,sinV,20*log10(abs(afr2)./max(max(abs(afr2)))));
        colormap(jet(256))
          caxis([-40, 0]);
        colorbar;
        view(2);
        xlabel('U');
        ylabel('V');
        %title('Reconstrução do feixe "objetivo"');
        shading interp;
         axis([-1 1 -1 1]);
        daspect([1 1 1])
```

APÊNDICE B – Códigos Desenvolvidos Para a Construção da Base de Dados Para Treino e Testes da CNN

```
clear;
close all;
clc;
PIXELS=11;%pixels no eixo x e y
%%
%variáveis padrão da antena

 PIXELS = 11;
 N = PIXELS; %Número de elementos em x.
 M = PIXELS; %Número de elementos em y.
 f = 12e9; % Freq de operação.
 c = 3e8; %Velocidade da luz.
 lambda = c/f; % Comprimento de onda.
 er = 4.5;%constante dielétrica.
 lambda_g = lambda/sqrt((er+1)/2); % Comprimento de onda guiado.
 dx = 0.0051;%lambda_g/3; %Espaçamento entre os elementos no eixo x
 dy = 0.0051;%lambda_g/3; %Espaçamento entre os elementos no eixo y.


l = 0.051;%largura da antena
lambda = c/f;%comprimento de onda espaço
k = 2*pi*f/c;%número de onda no espaço

%%
%coordenadas

theta0 = 0:pi/36:pi; % Elavação
phi0 = 0:pi/18:2*pi; % Azimute
[phi,theta] = meshgrid(phi0,theta0); % Malha nas coordenadas.
sinU = sin(theta).*cos(phi); % U-sine space.
sinV = sin(theta).*sin(phi); % V-sine space.


%%
% Posicionamento das Antenas

Px = -(N-1)/2:1:(N-1)/2; % Posição de cada antena no eixo x.
Py = -(-(M-1)/2:1:(M-1)/2); % Posição de cada antena no eixo y.
Px3d = dx*meshgrid(Px); % Matriz de posições.
Py3d = dy*meshgrid(Py)'; % Matriz de posições.


%%
%dataset creation

rho = 1; % Distância radial do ponto a ser projetado


thetav = 60; % de 0 a 60, com passo de 0.5°
phiv = 360; % de 0 a 180, com passo de 0.5°
NF = 3;

dtheta = 2*pi/(1.01*k*l + 1); %passo de theta

%%
```

```
%leitura h

H_HFSS = table2array(readtable('h.csv')); % Importa os dados da antena simualada.
H_HFSS(isnan(H_HFSS)) = 0;

Hy_HFSS = rot90((reshape(H_HFSS(:,4),N,M)-1i*reshape(H_HFSS(:,5),N,M))');
Hx_HFSS = rot90((reshape(H_HFSS(:,6),N,M)-1i*reshape(H_HFSS(:,7),N,M))');
Hz_HFSS = rot90((reshape(H_HFSS(:,8),N,M)-1i*reshape(H_HFSS(:,9),N,M))');
magH = abs(sqrt(Hx_HFSS.^2+Hy_HFSS.^2+Hz_HFSS.^2));

Href = 180*angle(Hy_HFSS)/pi;

%%
%loop
%variavel aleatoria que divide feixes de teste e de treino
%rande=randsrc(NF*thetav,phiv,[1 0;.75 .25]); % 3*60 - 3 da quantidade de fases↙
diferentes e 60 de theta ir de 0 a 60, 180 de phi ir de 0 a 180



Train_holo_Linha_1 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_2 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_3 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_4 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_5 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_6 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_7 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_8 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_9 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_10 = zeros(thetav*phiv*NF, 1);
Train_holo_Linha_11 = zeros(thetav*phiv*NF, 1);
Train_Beam = zeros(thetav*phiv*NF,37,37,1);



for i = 1:N
    factor(i,:) = 2^(i-1);
end

tst = 1;
trn= 1;

%%
phase_threshold = 20; % Limiar do módulo dasfases a serem consideradas.

%%
for thre = 1:NF
        phase_threshold = phase_threshold+5;
    for c = 1:thetav

        for d = 1:phiv

            %angulos de apontamento
            theta_p = (c-1)*1;
```

```
            phi_p = (d-1)*1;

            x0 = rho*sind(theta_p)*cosd(phi_p); % Coordenada X do ponto a ser projetado.
            y0 = rho*sind(theta_p)*sind(phi_p); % Coordenada Y do ponto a ser projetado.
            z0 = rho*cosd(theta_p); % Coordenada Z do ponto a ser projetado.
            r_linha = [x0 y0 z0]; % Coordenadas do ponto a ser projetado.

            %calculo bp
            beam_projection = zeros(N,M); % Criação do array da projeção.

            for n = 1:N
                for m = 1:M
                    % Cálculo da projeção sobre o arranjo usando onda esférica.
                    beam_projection(n,m) = exp(1i*(2*pi/lambda)*norm([Px3d(n,m) Py3d(n,m) 0]-r_linha))/norm([Px3d(n,m) Py3d(n,m) 0]-r_linha);
                end
            end

            beam_phase = angle(beam_projection)*180/pi;% Cálculo da fase da projeção sobre o arranjo.



            %comparação ondas

            Hologram_phase_2 = beam_phase-Href; % Gravação da fase do holograma.

            Holo2 = zeros(N,M); % Criação do array do holograma usando projeção.
            n_ant_2 = 0;
           for n = 1:1:N
               for m = 1:1:M
                   if abs(Hologram_phase_2(n,m)) <= phase_threshold  % Verificação se esta dentor do limiar.
                        Holo2(n,m) = 1; % Ligando a antena no holograma.
                        n_ant_2 = n_ant_2+1;
                   else
                        continue;
                   end
               end
           end
            Holo2(ceil(N/2),ceil(M/2)) = 0;

        %%
        %AF
        Sreal2 = (Holo2).*exp(1i*angle(Hy_HFSS)); %Matriz de excitação usando a fase da cavidade e a máscara do holograma.

        afr2 = zeros(length(theta0),length(phi0)); % Inicio do vetor do AFr.
        for n = 1:1:N
            for m =1:1:M
                afr2 =afr2+Sreal2(n,m).*exp((1i*2*pi/lambda)*(Px3d(n,m)*sinU+Py3d(n,m)*sinV));% Cálculo do fator de arranjo.
            end
```

```
        end


            md = abs(afr2)./max(max(abs(afr2)));

%             if(rande(thre*c,d) == 1)
                lb = reshape(Holo2, 1, []);
                int_Linha_1 = lb(1, 1:11)*factor;
            int_Linha_2 = lb(1, 12:22)*factor;
            int_Linha_3 = lb(1, 23:33)*factor;
            int_Linha_4 = lb(1, 34:44)*factor;
            int_Linha_5 = lb(1, 45:55)*factor;
            int_Linha_6 = lb(1, 56:66)*factor;
            int_Linha_7 = lb(1, 67:77)*factor;
            int_Linha_8 = lb(1, 78:88)*factor;
            int_Linha_9 = lb(1, 89:99)*factor;
            int_Linha_10 = lb(1, 100:110)*factor;
            int_Linha_11 = lb(1, 111:121)*factor;


                Train_holo_Linha_1(trn,:,:,1) = int_Linha_1;
                Train_holo_Linha_2(trn,:,:,1) = int_Linha_2;
                Train_holo_Linha_3(trn,:,:,1) = int_Linha_3;
                Train_holo_Linha_4(trn,:,:,1) = int_Linha_4;
                Train_holo_Linha_5(trn,:,:,1) = int_Linha_5;
                Train_holo_Linha_6(trn,:,:,1) = int_Linha_6;
                Train_holo_Linha_7(trn,:,:,1) = int_Linha_7;
                Train_holo_Linha_8(trn,:,:,1) = int_Linha_8;
                Train_holo_Linha_9(trn,:,:,1) = int_Linha_9;
                Train_holo_Linha_10(trn,:,:,1) = int_Linha_10;
                Train_holo_Linha_11(trn,:,:,1) = int_Linha_11;
                Train_Beam(trn, :, :, 1) = md;
                %trainhl(trn, :, :, 1)= Holo2;
                trn = trn + 1;


        end
    end
end

Angp = angle(beam_projection);
imagesc(Angp*180/pi);
%title('angulo de p','FontSize',9);
colormap(jet(256));
axis equal tight;
colorbar;
figure;

imagesc(md);
%title('angulo de p','FontSize',9);

axis equal tight;
colorbar;
figure;
```

```
colormap(gray(255));
image(Holo2, 'CDataMapping','scaled');
colorbar;
axis equal tight;


%%
%salvando arquivos de feixe

 %thisfilename = sprintf('Dataset/Hologramas/Teste/BP/Test_Beam.mat');
 %save(thisfilename, 'Test_Beam')
 thisfilename = sprintf('Dataset/Hologramas/Treino/BP/Train_Beam.mat');
 save(thisfilename,'Train_Beam');


 %salvando arquivos de holograma
 %1
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_1.mat');
 save(thisfilename,'Train_holo_Linha_1');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_1.mat');
 %save(thisfilename,'Test_Holo_Linha_1');
%2
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_2.mat');
 save(thisfilename,'Train_holo_Linha_2');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_2.mat');
 %save(thisfilename,'Test_Holo_Linha_2');
%3
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_3.mat');
 save(thisfilename,'Train_holo_Linha_3');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_3.mat');
 %save(thisfilename,'Test_Holo_Linha_3');
%4
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_4.mat');
 save(thisfilename,'Train_holo_Linha_4');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_4.mat');
 %save(thisfilename,'Test_Holo_Linha_4');
%5
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_5.mat');
 save(thisfilename,'Train_holo_Linha_5');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_5.mat');
 %save(thisfilename,'Test_Holo_Linha_5');
%6
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_6.mat');
 save(thisfilename,'Train_holo_Linha_6');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_6.mat');
 %save(thisfilename,'Test_Holo_Linha_6');
%7
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_7.mat');
 save(thisfilename,'Train_holo_Linha_7');
```

```
 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_7.mat');
 %save(thisfilename,'Test_Holo_Linha_7');
%8
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_8.mat');
 save(thisfilename,'Train_holo_Linha_8');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_8.mat');
 %save(thisfilename,'Test_Holo_Linha_8');
%9
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_9.mat');
 save(thisfilename,'Train_holo_Linha_9');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_9.mat');
 %save(thisfilename,'Test_Holo_Linha_9');
%10
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_10.mat');
 save(thisfilename,'Train_holo_Linha_10');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_10.mat');
 %save(thisfilename,'Test_Holo_Linha_10');
%11
 thisfilename = sprintf('Dataset/Hologramas/Treino/HOLO/Train_holo_Linha_11.mat');
 save(thisfilename,'Train_holo_Linha_11');

 %thisfilename = sprintf('Dataset/Hologramas/Teste/HOLO/Test_Holo_Linha_11.mat');
% save(thisfilename,'Test_Holo_Linha_11');
```

APÊNDICE C – Códigos Desenvolvidos Para a CNN

Bibliotecas

In [3]:
```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
import scipy.io
import csv
import pandas

from sklearn.metrics import confusion_matrix , classification_report
from sklearn.model_selection import train_test_split
```

Import Dataset

In [4]:
```
#Importando feixes
xtreino = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
xteste = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#importando apontamento
xteste_apontamento = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset
```

In [5]:
```
#importando todas as linhas da matriz
#1
ytreino1 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste1 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#2
ytreino2 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste2 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#3
ytreino3 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste3 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#4
ytreino4 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste4 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#5
ytreino5 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste5 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#6
ytreino6 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste6 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#7
ytreino7 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste7 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#8
ytreino8 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste8 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#9
ytreino9 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
yteste9 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#10
ytreino10 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Hologram
yteste10 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
#11
ytreino11 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Hologram
yteste11 = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Holograma
```

In [6]:
```
qtde = len(yteste1['Test_Holo_Linha_1'])
y_test = np.zeros([qtde,11, 1])
qtde_treino = len(ytreino1['Train_holo_Linha_1'])
```

```
y_train = np.zeros([qtde_treino,11, 1])
#1
y_train[:,0] = ytreino1['Train_holo_Linha_1']
y_test[:,0] = yteste1['Test_Holo_Linha_1']
#2
y_train[:,1] = ytreino2['Train_holo_Linha_2']
y_test[:,1]  = yteste2['Test_Holo_Linha_2']
#3
y_train[:,2] = ytreino3['Train_holo_Linha_3']
y_test[:,2] = yteste3['Test_Holo_Linha_3']
#4
y_train[:,3] = ytreino4['Train_holo_Linha_4']
y_test[:,3] = yteste4['Test_Holo_Linha_4']
#5
y_train[:,4] = ytreino5['Train_holo_Linha_5']
y_test[:,4] = yteste5['Test_Holo_Linha_5']
#6
y_train[:,5] = ytreino6['Train_holo_Linha_6']
y_test[:,5] = yteste6['Test_Holo_Linha_6']
#7
y_train[:,6] = ytreino7['Train_holo_Linha_7']
y_test[:,6] = yteste7['Test_Holo_Linha_7']
#8
y_train[:,7] = ytreino8['Train_holo_Linha_8']
y_test[:,7] = yteste8['Test_Holo_Linha_8']
#9
y_train[:,8] = ytreino9['Train_holo_Linha_9']
y_test[:,8] = yteste9['Test_Holo_Linha_9']
#10
y_train[:,9] = ytreino10['Train_holo_Linha_10']
y_test[:,9] = yteste10['Test_Holo_Linha_10']
#11
y_train[:,10] = ytreino11['Train_holo_Linha_11']
y_test[:,10] = yteste11['Test_Holo_Linha_11']
```

In [7]:
```
y_test = y_test.reshape(qtde,11,)
y_train = y_train.reshape(qtde_treino,11,)

y_test[:5,0]
```

Out[7]:
```
array([0., 0., 0., 0., 0.])
```

In [8]:
```
qtde
```

Out[8]:
```
8006
```

In [9]:
```
qtde_treino
```

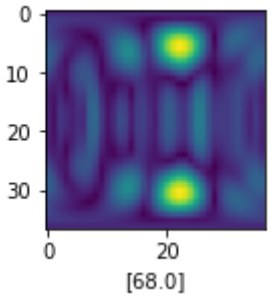
Out[9]:
```
64800
```

In [10]:
```
x_train = xtreino['Train_Beam']
x_test = xteste['Test_Beam']
x_test_apontamento = xteste_apontamento['Test_Apontamento']
pixelsx = 37
x_train.shape
```

Out[10]:
```
(64800, 37, 37)
```

In [11]:
```
def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel([y[index]])
```

In [12]:
```
plot_sample(x_train, y_train[:,0], 10300)
```



In [13]:
```
x_train[10300]
```

Out[13]:
```
array([[0.11770834, 0.11770834, 0.11770834, ..., 0.11770834, 0.11770834,
        0.11770834],
       [0.18519326, 0.18068698, 0.17507102, ..., 0.18677375, 0.18756657,
        0.18519326],
       [0.212157  , 0.18962309, 0.16741577, ..., 0.24689434, 0.23233061,
        0.212157  ],
       ...,
       [0.212157  , 0.18962309, 0.16741577, ..., 0.24689434, 0.23233061,
        0.212157  ],
       [0.18519326, 0.18068698, 0.17507102, ..., 0.18677375, 0.18756657,
        0.18519326],
       [0.11770834, 0.11770834, 0.11770834, ..., 0.11770834, 0.11770834,
        0.11770834]])
```

Model 1:

In [14]:
```
model1 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model1.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model1.add(layers.MaxPooling2D())
model1.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model1.add(layers.MaxPooling2D())
model1.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model1.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model1.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model1.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model1.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model1.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model1.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model1.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 35, 35, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 17, 17, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 15, 15, 16)        2320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 16)          0
```

```
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 16)          272       
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 16)          272       
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 16)          272       
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 16)          272       
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 16)          272       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 16)          0         
_________________________________________________________________
flatten (Flatten)            (None, 144)               0         
_________________________________________________________________
dense (Dense)                (None, 2048)              296960    
=================================================================
Total params: 300,800
Trainable params: 300,800
Non-trainable params: 0
_________________________________________________________________
```

Organizando dataset

In [15]:
```
x_train = x_train.reshape(-1, pixelsx, pixelsx, 1)
x_test = x_test.reshape(-1, pixelsx, pixelsx, 1)
```

In [16]:
```
# Split into training and validation dataset
x_train_1, x_val_1, y_train_1, y_val_1 = train_test_split(x_train, y_train[:,0], tes
x_train_2, x_val_2, y_train_2, y_val_2 = train_test_split(x_train, y_train[:,1], tes
x_train_3, x_val_3, y_train_3, y_val_3 = train_test_split(x_train, y_train[:,2], tes
x_train_4, x_val_4, y_train_4, y_val_4 = train_test_split(x_train, y_train[:,3], tes
x_train_5, x_val_5, y_train_5, y_val_5 = train_test_split(x_train, y_train[:,4], tes
x_train_6, x_val_6, y_train_6, y_val_6 = train_test_split(x_train, y_train[:,5], tes
x_train_7, x_val_7, y_train_7, y_val_7 = train_test_split(x_train, y_train[:,6], tes
x_train_8, x_val_8, y_train_8, y_val_8 = train_test_split(x_train, y_train[:,7], tes
x_train_9, x_val_9, y_train_9, y_val_9 = train_test_split(x_train, y_train[:,8], tes
x_train_10, x_val_10, y_train_10, y_val_10 = train_test_split(x_train, y_train[:,9],
x_train_11, x_val_11, y_train_11, y_val_11 = train_test_split(x_train, y_train[:,10]
```

Compilando e Avaliando Modelo

In [17]:
```
# Compile The Model
model1.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
# Fit And Evaluate The Model Using Validation Dataset
model1.fit(x_train_1, y_train_1, epochs=10, validation_data=(x_val_1, y_val_1))
model1.evaluate(x_test,y_test[:,0])
y_pred1 = model1.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 49s 28ms/step - loss: 3.8854 - accurac
y: 0.1903 - val_loss: 2.1986 - val_accuracy: 0.4219
Epoch 2/10
1620/1620 [==============================] - 41s 25ms/step - loss: 1.9445 - accurac
y: 0.4600 - val_loss: 1.3928 - val_accuracy: 0.5995
Epoch 3/10
1620/1620 [==============================] - 44s 27ms/step - loss: 1.2753 - accurac
y: 0.6155 - val_loss: 1.0705 - val_accuracy: 0.6767
Epoch 4/10
1620/1620 [==============================] - 42s 26ms/step - loss: 1.0100 - accurac
y: 0.6925 - val_loss: 0.9107 - val_accuracy: 0.7199
Epoch 5/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.8416 - accurac
```

```
y: 0.7374 - val_loss: 0.8611 - val_accuracy: 0.7290
Epoch 6/10
1620/1620 [==============================] - 41s 26ms/step - loss: 0.7625 - accurac
y: 0.7597 - val_loss: 0.8419 - val_accuracy: 0.7360
Epoch 7/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.6966 - accurac
y: 0.7770 - val_loss: 0.7043 - val_accuracy: 0.7778
Epoch 8/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.6318 - accurac
y: 0.7970 - val_loss: 0.6614 - val_accuracy: 0.7983
Epoch 9/10
1620/1620 [==============================] - 43s 26ms/step - loss: 0.5895 - accurac
y: 0.8084 - val_loss: 0.6171 - val_accuracy: 0.8138
Epoch 10/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.5636 - accurac
y: 0.8141 - val_loss: 0.6122 - val_accuracy: 0.8110
251/251 [==============================] - 2s 9ms/step - loss: 0.4678 - accuracy: 0.
8511
```

In [18]:

```
model2 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model2.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model2.add(layers.MaxPooling2D())
model2.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model2.add(layers.MaxPooling2D())
model2.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model2.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model2.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model2.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model2.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model2.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model2.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model2.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 35, 35, 16)        160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 16)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 15, 15, 16)        2320
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 16)          0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 3, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 144)               0
_________________________________________________________________
```

```
dense_1 (Dense)              (None, 2048)              296960
=================================================================
Total params: 300,800
Trainable params: 300,800
Non-trainable params: 0
_________________________________________________________________
```

In [19]:
```
model2.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f

# Fit And Evaluate The Model Using Validation Dataset
model2.fit(x_train_2, y_train_2, epochs=10, validation_data=(x_val_2, y_val_2))
model2.evaluate(x_test,y_test[:,1])
y_pred2 = model2.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 42s 25ms/step - loss: 3.4440 - accurac
y: 0.2377 - val_loss: 1.5357 - val_accuracy: 0.5649
Epoch 2/10
1620/1620 [==============================] - 44s 27ms/step - loss: 1.4094 - accurac
y: 0.5823 - val_loss: 1.1698 - val_accuracy: 0.6555
Epoch 3/10
1620/1620 [==============================] - 44s 27ms/step - loss: 1.0869 - accurac
y: 0.6724 - val_loss: 1.0317 - val_accuracy: 0.6928
Epoch 4/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.9186 - accurac
y: 0.7212 - val_loss: 0.8554 - val_accuracy: 0.7401
Epoch 5/10
1620/1620 [==============================] - 45s 28ms/step - loss: 0.7950 - accurac
y: 0.7535 - val_loss: 0.7439 - val_accuracy: 0.7741
Epoch 6/10
1620/1620 [==============================] - 46s 28ms/step - loss: 0.7150 - accurac
y: 0.7781 - val_loss: 0.7105 - val_accuracy: 0.7792
Epoch 7/10
1620/1620 [==============================] - 48s 30ms/step - loss: 0.6472 - accurac
y: 0.7941 - val_loss: 0.6534 - val_accuracy: 0.7971
Epoch 8/10
1620/1620 [==============================] - 46s 28ms/step - loss: 0.5945 - accurac
y: 0.8113 - val_loss: 0.6742 - val_accuracy: 0.7910
Epoch 9/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.5436 - accurac
y: 0.8266 - val_loss: 0.5678 - val_accuracy: 0.8227
Epoch 10/10
1620/1620 [==============================] - 47s 29ms/step - loss: 0.5106 - accurac
y: 0.8361 - val_loss: 0.5477 - val_accuracy: 0.8276
251/251 [==============================] - 3s 11ms/step - loss: 0.3878 - accuracy:
0.8629
```

In [20]:
```
model3 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model3.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model3.add(layers.MaxPooling2D())
model3.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model3.add(layers.MaxPooling2D())
model3.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model3.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model3.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model3.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model3.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model3.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model3.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model3.add(layers.Dense(2048, activation='softmax'))
```

```
# Show summary of the model
#model.summary()
```

In [21]:
```
model3.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
# Fit And Evaluate The Model Using Validation Dataset
model3.fit(x_train_3, y_train_3, epochs=10, validation_data=(x_val_3, y_val_3))
model3.evaluate(x_test,y_test[:,2])
y_pred3 = model3.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 47s 29ms/step - loss: 3.3465 - accurac
y: 0.2715 - val_loss: 1.3517 - val_accuracy: 0.5964
Epoch 2/10
1620/1620 [==============================] - 44s 27ms/step - loss: 1.2097 - accurac
y: 0.6430 - val_loss: 0.9291 - val_accuracy: 0.7183
Epoch 3/10
1620/1620 [==============================] - 48s 30ms/step - loss: 0.8976 - accurac
y: 0.7227 - val_loss: 0.7979 - val_accuracy: 0.7559
Epoch 4/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.7524 - accurac
y: 0.7639 - val_loss: 0.6966 - val_accuracy: 0.7823
Epoch 5/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.6535 - accurac
y: 0.7894 - val_loss: 0.6278 - val_accuracy: 0.7990
Epoch 6/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.5886 - accurac
y: 0.8093 - val_loss: 0.5485 - val_accuracy: 0.8247
Epoch 7/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.5369 - accurac
y: 0.8244 - val_loss: 0.5363 - val_accuracy: 0.8275
Epoch 8/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.4977 - accurac
y: 0.8363 - val_loss: 0.5325 - val_accuracy: 0.8270
Epoch 9/10
1620/1620 [==============================] - 41s 25ms/step - loss: 0.4679 - accurac
y: 0.8448 - val_loss: 0.4886 - val_accuracy: 0.8415
Epoch 10/10
1620/1620 [==============================] - 41s 26ms/step - loss: 0.4293 - accurac
y: 0.8585 - val_loss: 0.4487 - val_accuracy: 0.8543
251/251 [==============================] - 2s 9ms/step - loss: 0.3437 - accuracy: 0.
8796
```

In [22]:
```
model4 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model4.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model4.add(layers.MaxPooling2D())
model4.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model4.add(layers.MaxPooling2D())
model4.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model4.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model4.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model4.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model4.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model4.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model4.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model4.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
model4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_21 (Conv2D)           (None, 35, 35, 16)        160
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 17, 17, 16)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 15, 15, 16)        2320
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 7, 7, 16)          0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 3, 3, 16)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 144)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2048)              296960
=================================================================
Total params: 300,800
Trainable params: 300,800
Non-trainable params: 0
_________________________________________________________________
```

In [23]:

```
model4.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
# Fit And Evaluate The Model Using Validation Dataset
model4.fit(x_train_4, y_train_4, epochs=10, validation_data=(x_val_4, y_val_4))
model4.evaluate(x_test,y_test[:,3])
y_pred4 = model4.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 41s 25ms/step - loss: 3.1004 - accurac
y: 0.3122 - val_loss: 1.2366 - val_accuracy: 0.6219
Epoch 2/10
1620/1620 [==============================] - 43s 26ms/step - loss: 1.1280 - accurac
y: 0.6490 - val_loss: 0.9379 - val_accuracy: 0.7005
Epoch 3/10
1620/1620 [==============================] - 47s 29ms/step - loss: 0.8538 - accurac
y: 0.7283 - val_loss: 0.7657 - val_accuracy: 0.7569
Epoch 4/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.7360 - accurac
y: 0.7621 - val_loss: 0.7061 - val_accuracy: 0.7735
Epoch 5/10
1620/1620 [==============================] - 39s 24ms/step - loss: 0.6405 - accurac
y: 0.7907 - val_loss: 0.6393 - val_accuracy: 0.7894
Epoch 6/10
1620/1620 [==============================] - 39s 24ms/step - loss: 0.5885 - accurac
y: 0.8075 - val_loss: 0.5773 - val_accuracy: 0.8136
Epoch 7/10
1620/1620 [==============================] - 40s 24ms/step - loss: 0.5350 - accurac
y: 0.8232 - val_loss: 0.5422 - val_accuracy: 0.8148
Epoch 8/10
1620/1620 [==============================] - 39s 24ms/step - loss: 0.5111 - accurac
y: 0.8295 - val_loss: 0.5053 - val_accuracy: 0.8328
Epoch 9/10
1620/1620 [==============================] - 39s 24ms/step - loss: 0.4761 - accurac
y: 0.8397 - val_loss: 0.4620 - val_accuracy: 0.8503
Epoch 10/10
1620/1620 [==============================] - 52s 32ms/step - loss: 0.4530 - accurac
y: 0.8447 - val_loss: 0.4218 - val_accuracy: 0.8643
251/251 [==============================] - 6s 20ms/step - loss: 0.3620 - accuracy:
0.8830
```

In [24]:
```
model5 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model5.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model5.add(layers.MaxPooling2D())
model5.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model5.add(layers.MaxPooling2D())
model5.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model5.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model5.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model5.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model5.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model5.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model5.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model5.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
#model.summary()
```

In [25]:
```
model5.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
# Fit And Evaluate The Model Using Validation Dataset
model5.fit(x_train_5, y_train_5, epochs=10, validation_data=(x_val_5, y_val_5))
model5.evaluate(x_test,y_test[:,4])
y_pred5 = model5.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 41s 24ms/step - loss: 2.7648 - accurac
y: 0.3596 - val_loss: 1.1265 - val_accuracy: 0.6343
Epoch 2/10
1620/1620 [==============================] - 38s 24ms/step - loss: 1.0166 - accurac
y: 0.6731 - val_loss: 0.8210 - val_accuracy: 0.7329
Epoch 3/10
1620/1620 [==============================] - 38s 24ms/step - loss: 0.7733 - accurac
y: 0.7423 - val_loss: 0.7119 - val_accuracy: 0.7580
Epoch 4/10
1620/1620 [==============================] - 40s 25ms/step - loss: 0.6645 - accurac
y: 0.7741 - val_loss: 0.6202 - val_accuracy: 0.7918
Epoch 5/10
1620/1620 [==============================] - 48s 29ms/step - loss: 0.6049 - accurac
y: 0.7909 - val_loss: 0.6125 - val_accuracy: 0.7947
Epoch 6/10
1620/1620 [==============================] - 43s 26ms/step - loss: 0.5650 - accurac
y: 0.8008 - val_loss: 0.5905 - val_accuracy: 0.7935
Epoch 7/10
1620/1620 [==============================] - 40s 25ms/step - loss: 0.5269 - accurac
y: 0.8159 - val_loss: 0.5157 - val_accuracy: 0.8209
Epoch 8/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.5075 - accurac
y: 0.8225 - val_loss: 0.5293 - val_accuracy: 0.8182
Epoch 9/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.4760 - accurac
y: 0.8329 - val_loss: 0.4870 - val_accuracy: 0.8325
Epoch 10/10
1620/1620 [==============================] - 47s 29ms/step - loss: 0.4509 - accurac
y: 0.8393 - val_loss: 0.4556 - val_accuracy: 0.8400
251/251 [==============================] - 3s 11ms/step - loss: 0.3782 - accuracy:
0.8681
```

In [26]:
```
model6 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model6.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model6.add(layers.MaxPooling2D())
```

```
model6.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model6.add(layers.MaxPooling2D())
model6.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model6.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model6.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model6.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model6.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model6.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model6.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model6.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
model6.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_35 (Conv2D)           (None, 35, 35, 16)        160
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 17, 17, 16)        0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 15, 15, 16)        2320
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 7, 7, 16)          0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 3, 3, 16)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 144)               0
_________________________________________________________________
dense_5 (Dense)              (None, 2048)              296960
=================================================================
Total params: 300,800
Trainable params: 300,800
Non-trainable params: 0
_________________________________________________________________
```

In [27]:

```
model6.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
# Fit And Evaluate The Model Using Validation Dataset
model6.fit(x_train_6, y_train_6, epochs=10, validation_data=(x_val_6, y_val_6))
model6.evaluate(x_test,y_test[:,5])
y_pred6 = model6.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 47s 28ms/step - loss: 2.1614 - accurac
y: 0.4558 - val_loss: 0.8295 - val_accuracy: 0.7025
Epoch 2/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.7677 - accurac
y: 0.7176 - val_loss: 0.6102 - val_accuracy: 0.7736
Epoch 3/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.5990 - accurac
y: 0.7765 - val_loss: 0.5165 - val_accuracy: 0.8046
Epoch 4/10
```

```
1620/1620 [==============================] - 42s 26ms/step - loss: 0.5134 - accurac
y: 0.8088 - val_loss: 0.4758 - val_accuracy: 0.8216
Epoch 5/10
1620/1620 [==============================] - 45s 28ms/step - loss: 0.4703 - accurac
y: 0.8224 - val_loss: 0.4663 - val_accuracy: 0.8225
Epoch 6/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.4235 - accurac
y: 0.8395 - val_loss: 0.3764 - val_accuracy: 0.8610
Epoch 7/10
1620/1620 [==============================] - 45s 28ms/step - loss: 0.4121 - accurac
y: 0.8446 - val_loss: 0.3884 - val_accuracy: 0.8485
Epoch 8/10
1620/1620 [==============================] - 45s 28ms/step - loss: 0.3769 - accurac
y: 0.8583 - val_loss: 0.3202 - val_accuracy: 0.8859
Epoch 9/10
1620/1620 [==============================] - 49s 30ms/step - loss: 0.3621 - accurac
y: 0.8644 - val_loss: 0.3835 - val_accuracy: 0.8596
Epoch 10/10
1620/1620 [==============================] - 47s 29ms/step - loss: 0.3418 - accurac
y: 0.8728 - val_loss: 0.3594 - val_accuracy: 0.8626
251/251 [==============================] - 2s 9ms/step - loss: 0.3255 - accuracy: 0.
8763
```

In [28]:

```
model7 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model7.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model7.add(layers.MaxPooling2D())
model7.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model7.add(layers.MaxPooling2D())
model7.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model7.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model7.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model7.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model7.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model7.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model7.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model7.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
#model.summary()
```

In [29]:

```
model7.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
# Fit And Evaluate The Model Using Validation Dataset
model7.fit(x_train_7, y_train_7, epochs=10, validation_data=(x_val_7, y_val_7))
model7.evaluate(x_test,y_test[:,6])
y_pred7 = model7.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 42s 26ms/step - loss: 2.9186 - accurac
y: 0.3455 - val_loss: 0.9591 - val_accuracy: 0.6880
Epoch 2/10
1620/1620 [==============================] - 40s 25ms/step - loss: 0.8390 - accurac
y: 0.7236 - val_loss: 0.6888 - val_accuracy: 0.7658
Epoch 3/10
1620/1620 [==============================] - 41s 25ms/step - loss: 0.6067 - accurac
y: 0.7900 - val_loss: 0.5415 - val_accuracy: 0.8186
Epoch 4/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.5185 - accurac
y: 0.8193 - val_loss: 0.5174 - val_accuracy: 0.8215
Epoch 5/10
1620/1620 [==============================] - 41s 25ms/step - loss: 0.4516 - accurac
y: 0.8415 - val_loss: 0.4377 - val_accuracy: 0.8473
```

```
Epoch 6/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.4220 - accurac
y: 0.8508 - val_loss: 0.4246 - val_accuracy: 0.8523
Epoch 7/10
1620/1620 [==============================] - 41s 25ms/step - loss: 0.3861 - accurac
y: 0.8653 - val_loss: 0.3854 - val_accuracy: 0.8653
Epoch 8/10
1620/1620 [==============================] - 48s 30ms/step - loss: 0.3568 - accurac
y: 0.8740 - val_loss: 0.3586 - val_accuracy: 0.8791
Epoch 9/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.3354 - accurac
y: 0.8813 - val_loss: 0.3411 - val_accuracy: 0.8832
Epoch 10/10
1620/1620 [==============================] - 46s 29ms/step - loss: 0.3173 - accurac
y: 0.8866 - val_loss: 0.3500 - val_accuracy: 0.8837
251/251 [==============================] - 3s 10ms/step - loss: 0.2776 - accuracy:
0.9031
```

In [30]:

```
model8 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model8.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model8.add(layers.MaxPooling2D())
model8.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model8.add(layers.MaxPooling2D())
model8.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model8.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model8.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model8.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model8.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model8.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model8.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model8.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
model8.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_49 (Conv2D)           (None, 35, 35, 16)        160
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 17, 17, 16)        0
_________________________________________________________________
conv2d_50 (Conv2D)           (None, 15, 15, 16)        2320
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 7, 7, 16)          0
_________________________________________________________________
conv2d_51 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_52 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_53 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_54 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_55 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 3, 3, 16)          0
_________________________________________________________________
flatten_7 (Flatten)          (None, 144)               0
_________________________________________________________________
dense_7 (Dense)              (None, 2048)              296960
```

```
=================================================================
Total params: 300,800
Trainable params: 300,800
Non-trainable params: 0
_________________________________________________________________
```

In [31]:
```
model8.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
# Fit And Evaluate The Model Using Validation Dataset
model8.fit(x_train_8, y_train_8, epochs=10, validation_data=(x_val_8, y_val_8))
model8.evaluate(x_test,y_test[:,7])
y_pred8 = model8.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 44s 27ms/step - loss: 3.1333 - accurac
y: 0.3171 - val_loss: 1.2520 - val_accuracy: 0.6265
Epoch 2/10
1620/1620 [==============================] - 44s 27ms/step - loss: 1.1439 - accurac
y: 0.6481 - val_loss: 0.9896 - val_accuracy: 0.6917
Epoch 3/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.9194 - accurac
y: 0.7047 - val_loss: 0.8615 - val_accuracy: 0.7262
Epoch 4/10
1620/1620 [==============================] - 41s 26ms/step - loss: 0.7846 - accurac
y: 0.7446 - val_loss: 0.7600 - val_accuracy: 0.7480
Epoch 5/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.7004 - accurac
y: 0.7683 - val_loss: 0.6737 - val_accuracy: 0.7838
Epoch 6/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.6361 - accurac
y: 0.7877 - val_loss: 0.6045 - val_accuracy: 0.7998
Epoch 7/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.5795 - accurac
y: 0.8070 - val_loss: 0.6238 - val_accuracy: 0.7898
Epoch 8/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.5453 - accurac
y: 0.8170 - val_loss: 0.5316 - val_accuracy: 0.8262
Epoch 9/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.5050 - accurac
y: 0.8300 - val_loss: 0.5180 - val_accuracy: 0.8283
Epoch 10/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.4860 - accurac
y: 0.8388 - val_loss: 0.4722 - val_accuracy: 0.8438
251/251 [==============================] - 3s 10ms/step - loss: 0.3986 - accuracy:
0.8646
```

In [32]:
```
model9 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model9.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_sh
model9.add(layers.MaxPooling2D())
model9.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model9.add(layers.MaxPooling2D())
model9.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model9.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model9.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model9.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model9.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model9.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model9.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model9.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
#model.summary()
```

In [33]:
```
model9.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
# Fit And Evaluate The Model Using Validation Dataset
model9.fit(x_train_9, y_train_9, epochs=10, validation_data=(x_val_9, y_val_9))
model9.evaluate(x_test,y_test[:,8])
y_pred9 = model9.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 43s 26ms/step - loss: 3.3783 - accurac
y: 0.2673 - val_loss: 1.1325 - val_accuracy: 0.6609
Epoch 2/10
1620/1620 [==============================] - 40s 25ms/step - loss: 0.9944 - accurac
y: 0.6934 - val_loss: 0.7670 - val_accuracy: 0.7518
Epoch 3/10
1620/1620 [==============================] - 40s 25ms/step - loss: 0.7420 - accurac
y: 0.7565 - val_loss: 0.6442 - val_accuracy: 0.7948
Epoch 4/10
1620/1620 [==============================] - 41s 26ms/step - loss: 0.6078 - accurac
y: 0.7998 - val_loss: 0.5512 - val_accuracy: 0.8233
Epoch 5/10
1620/1620 [==============================] - 41s 25ms/step - loss: 0.5358 - accurac
y: 0.8230 - val_loss: 0.4900 - val_accuracy: 0.8464
Epoch 6/10
1620/1620 [==============================] - 40s 25ms/step - loss: 0.4737 - accurac
y: 0.8403 - val_loss: 0.4619 - val_accuracy: 0.8513
Epoch 7/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.4505 - accurac
y: 0.8509 - val_loss: 0.4237 - val_accuracy: 0.8639
Epoch 8/10
1620/1620 [==============================] - 41s 25ms/step - loss: 0.4038 - accurac
y: 0.8665 - val_loss: 0.4697 - val_accuracy: 0.8508
Epoch 9/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.3797 - accurac
y: 0.8736 - val_loss: 0.3819 - val_accuracy: 0.8782
Epoch 10/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.3492 - accurac
y: 0.8827 - val_loss: 0.3601 - val_accuracy: 0.8819
251/251 [==============================] - 3s 10ms/step - loss: 0.2925 - accuracy:
0.9014
```

In [34]:
```
model10 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
model10.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_s
model10.add(layers.MaxPooling2D())
model10.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model10.add(layers.MaxPooling2D())
model10.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model10.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model10.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model10.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model10.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model10.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model10.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model10.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
model10.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_63 (Conv2D)           (None, 35, 35, 16)        160
```

```
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 17, 17, 16)        0
_________________________________________________________________
conv2d_64 (Conv2D)           (None, 15, 15, 16)        2320
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 7, 7, 16)          0
_________________________________________________________________
conv2d_65 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_66 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_67 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_68 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
conv2d_69 (Conv2D)           (None, 7, 7, 16)          272
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 3, 3, 16)          0
_________________________________________________________________
flatten_9 (Flatten)          (None, 144)               0
_________________________________________________________________
dense_9 (Dense)              (None, 2048)              296960
=================================================================
Total params: 300,800
Trainable params: 300,800
Non-trainable params: 0
_________________________________________________________________
```

In [35]:
```
model10.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(
# Fit And Evaluate The Model Using Validation Dataset
model10.fit(x_train_10, y_train_10, epochs=10, validation_data=(x_val_10, y_val_10))
model10.evaluate(x_test,y_test[:,9])
y_pred10 = model10.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 42s 26ms/step - loss: 3.6924 - accurac
y: 0.2050 - val_loss: 1.9658 - val_accuracy: 0.4457
Epoch 2/10
1620/1620 [==============================] - 41s 25ms/step - loss: 1.7728 - accurac
y: 0.4913 - val_loss: 1.4049 - val_accuracy: 0.5910
Epoch 3/10
1620/1620 [==============================] - 42s 26ms/step - loss: 1.2990 - accurac
y: 0.6092 - val_loss: 1.1256 - val_accuracy: 0.6581
Epoch 4/10
1620/1620 [==============================] - 40s 24ms/step - loss: 1.0545 - accurac
y: 0.6782 - val_loss: 0.9477 - val_accuracy: 0.7046
Epoch 5/10
1620/1620 [==============================] - 40s 25ms/step - loss: 0.9056 - accurac
y: 0.7192 - val_loss: 0.8322 - val_accuracy: 0.7408
Epoch 6/10
1620/1620 [==============================] - 41s 25ms/step - loss: 0.8284 - accurac
y: 0.7381 - val_loss: 0.7624 - val_accuracy: 0.7572
Epoch 7/10
1620/1620 [==============================] - 43s 26ms/step - loss: 0.7559 - accurac
y: 0.7610 - val_loss: 0.7445 - val_accuracy: 0.7599
Epoch 8/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.6911 - accurac
y: 0.7809 - val_loss: 0.6530 - val_accuracy: 0.7906
Epoch 9/10
1620/1620 [==============================] - 43s 26ms/step - loss: 0.6299 - accurac
y: 0.7954 - val_loss: 0.6191 - val_accuracy: 0.8039
Epoch 10/10
1620/1620 [==============================] - 44s 27ms/step - loss: 0.6045 - accurac
y: 0.8037 - val_loss: 0.6013 - val_accuracy: 0.8061
251/251 [==============================] - 2s 9ms/step - loss: 0.4836 - accuracy: 0.
8340
```

In [36]:
```
model11 = models.Sequential()
# Feature Extraction Section (The Convolution and The Pooling Layer)
```

```
model11.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_s
model11.add(layers.MaxPooling2D())
model11.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model11.add(layers.MaxPooling2D())
model11.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model11.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model11.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model11.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
model11.add(layers.Conv2D(filters=16, kernel_size=(1, 1), activation='relu'))
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model11.add(layers.MaxPooling2D())
#model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
# Reshape the image into one-dimensional vector
model11.add(layers.Flatten())
# Classification Section (The Fully Connected Layer)
#model.add(layers.Dense(120, activation='relu'))
#model.add(layers.Dense(84, activation='relu'))
model11.add(layers.Dense(2048, activation='softmax'))
# Show summary of the model
#model.summary()
```

In [37]:
```
model11.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(
# Fit And Evaluate The Model Using Validation Dataset
model11.fit(x_train_11, y_train_11, epochs=10, validation_data=(x_val_11, y_val_11))
model11.evaluate(x_test,y_test[:,10])
y_pred11 = model11.predict(x_test)
```

```
Epoch 1/10
1620/1620 [==============================] - 43s 26ms/step - loss: 3.8551 - accurac
y: 0.2093 - val_loss: 2.2251 - val_accuracy: 0.4053
Epoch 2/10
1620/1620 [==============================] - 40s 25ms/step - loss: 1.9825 - accurac
y: 0.4578 - val_loss: 1.4877 - val_accuracy: 0.5766
Epoch 3/10
1620/1620 [==============================] - 40s 25ms/step - loss: 1.3603 - accurac
y: 0.6055 - val_loss: 1.1272 - val_accuracy: 0.6791
Epoch 4/10
1620/1620 [==============================] - 40s 25ms/step - loss: 1.0427 - accurac
y: 0.6898 - val_loss: 0.9844 - val_accuracy: 0.7117
Epoch 5/10
1620/1620 [==============================] - 43s 27ms/step - loss: 0.8991 - accurac
y: 0.7270 - val_loss: 0.8371 - val_accuracy: 0.7552
Epoch 6/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.7880 - accurac
y: 0.7560 - val_loss: 0.7619 - val_accuracy: 0.7721
Epoch 7/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.7113 - accurac
y: 0.7776 - val_loss: 0.7193 - val_accuracy: 0.7839
Epoch 8/10
1620/1620 [==============================] - 41s 26ms/step - loss: 0.6574 - accurac
y: 0.7961 - val_loss: 0.6543 - val_accuracy: 0.8046
Epoch 9/10
1620/1620 [==============================] - 42s 26ms/step - loss: 0.5813 - accurac
y: 0.8143 - val_loss: 0.6122 - val_accuracy: 0.8150
Epoch 10/10
1620/1620 [==============================] - 41s 25ms/step - loss: 0.5567 - accurac
y: 0.8211 - val_loss: 0.6088 - val_accuracy: 0.8141
251/251 [==============================] - 3s 9ms/step - loss: 0.4910 - accuracy: 0.
8399
```

In [38]:
```
y_predicted = np.zeros([qtde,11])
y_predicted[:,0] = [np.argmax(element) for element in y_pred1]
y_predicted[:,1] = [np.argmax(element) for element in y_pred2]
y_predicted[:,2] = [np.argmax(element) for element in y_pred3]
y_predicted[:,3] = [np.argmax(element) for element in y_pred4]
y_predicted[:,4] = [np.argmax(element) for element in y_pred5]
y_predicted[:,5] = [np.argmax(element) for element in y_pred6]
```

```
y_predicted[:,6] = [np.argmax(element) for element in y_pred7]
y_predicted[:,7] = [np.argmax(element) for element in y_pred8]
y_predicted[:,8] = [np.argmax(element) for element in y_pred9]
y_predicted[:,9] = [np.argmax(element) for element in y_pred10]
y_predicted[:,10] = [np.argmax(element) for element in y_pred11]
```

```
1029.0
```

Out[38]:
```
1029.0
```

Exportando Modelos

In [55]:
```
model1.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model1')
model2.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model2')
model3.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model3')
model4.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model4')
model5.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model5')
model6.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model6')
model7.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model7')
model8.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model8')
model9.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model9')
model10.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model10')
model11.save('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\Ant11\\Model11')
```

```
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model1\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model2\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model3\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model4\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model5\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model6\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model7\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model8\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model9\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model10\assets
INFO:tensorflow:Assets written to: C:\Users\wemer\Desktop\tcc\Arquivos\Models\Ant11
\Model11\assets
```

Testes únicos

In [56]:
```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
import scipy.io
import csv
import pandas

from sklearn.metrics import confusion_matrix , classification_report
from sklearn.model_selection import train_test_split

from tensorflow import keras
```

In [57]:
```
#Lendo redes neurais modo normal
```

```
model1 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model2 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model3 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model4 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model5 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model6 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model7 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model8 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model9 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\\
model10 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\
model11 = keras.models.load_model('C:\\Users\\wemer\\Desktop\\tcc\\Arquivos\\Models\
```

In [58]:
```
def conv2_binrow(y):
    b =  [int(i) for i in list('{0:0b}'.format(y))]
    c = np.asarray(b)
    d = np.pad(c, (11-len(c), 0), 'constant')
    return d
```

In [59]:
```
def linhas_cnn(X):
    X = X.reshape(-1, pixelsx, pixelsx, 1)
    y_pred = np.zeros([1,11])
    y_pred[:,0] = [np.argmax(element) for element in model1.predict(X)]
    y_pred[:,1] = [np.argmax(element) for element in model2.predict(X)]
    y_pred[:,2] = [np.argmax(element) for element in model3.predict(X)]
    y_pred[:,3] = [np.argmax(element) for element in model4.predict(X)]
    y_pred[:,4] = [np.argmax(element) for element in model5.predict(X)]
    y_pred[:,5] = [np.argmax(element) for element in model6.predict(X)]
    y_pred[:,6] = [np.argmax(element) for element in model7.predict(X)]
    y_pred[:,7] = [np.argmax(element) for element in model8.predict(X)]
    y_pred[:,8] = [np.argmax(element) for element in model9.predict(X)]
    y_pred[:,9] = [np.argmax(element) for element in model10.predict(X)]
    y_pred[:,10] = [np.argmax(element) for element in model11.predict(X)]
    return y_pred;
```

In [63]:
```
def convbin(Y):
    y_bin = np.zeros([11,11])
    for k in range(0,11):
        y_bin[:,k] = conv2_binrow(int(Y[k]))
    return y_bin;
```

Testes

In [91]:
```
#carregando feixe
x_opt_teste = scipy.io.loadmat('C:\\Users\\wemer\\Documents\\MATLAB\\Dataset\\Hologr
x_opt_test = x_opt_teste['opt_test']
pixelsx = 37
```

In [92]:
```
a = linhas_cnn(x_opt_test)
c = np.flipud(convbin(a[0]))
scipy.io.savemat('./Arquivos/Testes_Feixes/opt_test.mat', dict(x=2,y=c))
```

In [ ]:
```

```