UNIVERSIDADE ESTADUAL PAULISTA (UNESP)
"JÚLIO DE MESQUITA FILHO"
CAMPUS SÃO JOÃO DA BOA VISTA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LUCAS MARIM DA SILVA

Redes neurais convolucionais para predição de probabilidade de erro de bit em sistemas de comunicações ópticas coerentes digitais limitados por modulação de fase não linear

São João da Boa Vista/SP
2022

LUCAS MARIM DA SILVA

Redes neurais convolucionais para predição de probabilidade de erro de bit em sistemas de comunicações ópticas coerentes digitais limitados por modulação de fase não linear

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de São Paulo (Campus São João da Boa Vista) como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica pelo Programa de Pós-graduação em Engenharia Elétrica.

Área de concentração: Sistemas Eletrônicos

Orientador: Prof. Dr. Ivan Aritz Aldaya Garde

Coorientador: Prof. Dr. José Augusto de Oliveira

São João da Boa Vista/SP

2022

S586r

Silva, Lucas Marim da

Redes neurais convolucionais para predição de probabilidade de erro de bit em sistemas de comunicações ópticas coerentes digitais limitados por modulação de fase não linear / Lucas Marim da Silva. -- São João da Boa Vista, 2022

103 p.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Engenharia, São João da Boa Vista

Orientador: Ivan Aritz Aldaya Garde

Coorientador: José Augusto de Oliveira

1. Comunicações ópticas. 2. Redes neurais (Computação). 3. Telecomunicações. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Engenharia, São João da Boa Vista. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de São João da Boa Vista

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Redes neurais convolucionais para predição de probabilidade de erro de bit em sistemas de comunicações ópticas coerentes digitais limitados por modulação de fase não linear

AUTOR: LUCAS MARIM DA SILVA

ORIENTADOR: IVAN ARITZ ALDAYA GARDE

Aprovado como parte das exigências para obtenção do Título de Mestre em ENGENHARIA ELÉTRICA, área: Sistemas Eletrônicos pela Comissão Examinadora:

Prof. Dr. IVAN ARITZ ALDAYA GARDE (Participaçao Presencial)
Coordenadoria de Curso de Engenharia Eletronica e de Telecomunicacoes / Faculdade de Engenharia de Sao Joao da Boa Vista - UNESP

Prof. Dr. RAFAEL ABRANTES PENCHEL (Participaçao Presencial)
Coordenadoria de Curso de Engenharia Eletrônica e de Telecomunicações / Faculdade de Engenharia de São João da Boa Vista - UNESP

Prof. Dr. LUIZ HENRIQUE BONANI DO NASCIMENTO (Participaçao Presencial)
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas / Universidade Federal do ABC

São João da Boa Vista, 01 de junho de 2022

Maria Luiza Sarubi Barreto
Diretora Técnica Acadêmica

Dedico este trabalho ao meu avô, José Roberto Marim, que tão cedo nos deixou, mas que está eternizado em nossos corações.

Agradecimentos

À Deus, primeiramente, por ter me concedido o dom da vida e me dado forças para superar cada obstáculo, guiando-me em meio a passos incertos, protegendo-me dia após dia, dando-me forças nos momentos de dificuldade e me amando incondicionalmente.

À meus pais, Silvia e Joel, que forneceram o meu sustento durante toda minha vida, me motivando fielmente ao longo de minha jornada acadêmica, acreditando na minha capacidade e me apoiando durante as horas de cansaço.

À meus familiares, em especial meus irmãos Ana Carolina, Guilherme e Heloysa, que sempre foram grandes referências e fonte de inspiração, representando um porto seguro no qual sempre pude me firmar em momentos difíceis.

Aos meus orientadores, Prof. Dr. Ivan Aritz Aldaya Garde e Prof. Dr. José Augusto de Oliveira, que aceitaram o desafio de me acompanhar ao longo do desenvolvimento desta pesquisa de mestrado, provendo-me todo o apoio, suporte e confiança necessários para a elaboração deste trabalho.

À todos que direta ou indiretamente fizeram parte da minha formação e me apoiaram ao longo de minha graduação e mestrado.

"Se, porém, algum de vós necessita de sabedoria,
peça-a a Deus, que a todos dá liberalmente,
e nada lhes impropera; e ser-lhe-á concedida."
(Tiago 1:5)

Resumo

SILVA, Lucas Marim da. **Redes neurais convolucionais para predição de probabilidade de erro de bit em sistemas de comunicações ópticas coerentes digitais limitados por modulação de fase não linear**. 2022. 103 f. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica, Universidade Estadual Paulista, São Paulo, 2022.

Neste trabalho são desenvolvidas técnicas para estimar a probabilidade de erro de bit (BER) em sistemas de comunicações ópticas digitais coerentes utilizando redes neurais convolucionais (CNNs). A estimativa é performada por meio do processamento histogramas de constelações de sinais por um algoritmo de regressão, capaz de generalizar a estimativa para redes ópticas passivas (PONs) com diferentes comprimentos de enlace e valores de potência de transmissão. Os resultados revelam que, utilizando uma CNN capaz de processar histogramas compostos por 10.000 símbolos e 64 *bins*, o erro entre o valor médio de BER estimado e esperado foi igual ou inferior a 10.87% para uma PON de 150 km considerando a faixa de valores de potência em que o sistema é limitado por modulação de fase não linear. O custo computacional necessário para realizar uma estimativa de BER utilizando a CNN descrita é de $195,61 \times 10^6$ operações de ponto flutuante.

Palavras-chaves: Comunicações ópticas; Probabilidade de erro de bit; Ruído de fase não linear; Redes neurais convolucionais.

Abstract

Silva, Lucas Marim da. **Convolutional neural networks for bit error ratio prediction in digital coherent optical communication systems limited by non-linear phase modulation**. 2022. 103 p. Dissertation (Master of Science) – São Paulo State University (UNESP), Campus São João da Boa Vista, São João da Boa Vista, 2022.

In this work, we developed techniques to estimate bit error ratio (BER) in digital coherent optical communications systems using convolutional neural networks (CNNs). The estimation is performed by processing histograms of constellations diagrams considering a regression algorithm capable of generalizing the estimation to different passive optical networks (PONs) configurations. Results reveal that a CNN trained to process histograms of 64 *bins* composed by 10,000 symbols presents an estimation error equal to or less than 10.87% considering a 150 km PON for launch optical power values over which the system is limited by non-linear phase modulation. The computational cost required to perform a BER estimation using the described CNN is 195.61×10^6 floating point operations.

Keywords: Optical communications; Bit error probability; Nonlinear phase noise; Convolutional neural networks.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

AI - *Artificial intelligence*

ANN - *Artificial neural network*

ASE - *Amplified spontaneous emission*

BER - *Bit error rate*

CNN - *Convolutional neural network*

DBP - *Digital back propagation*

DCF - *Dispersion-compensating fiber*

EDFA - *Erbium doped fiber amplifiers*

FWM - *Four-wave mixing*

GVD - *Group-velocity dispersion*

IVSTF - *Inverse Volterra Series Transfer Function*

LOP - *Launch optical power*

LR-PON - *Long reach passive optical network*

MLP - *Multilayer perceptron*

OOK - *On-off keying*

PC - *Principal component*

PCA - *Principal component analysis*

PMD - *Polarization mode dispersion*

PON - *Passive optical network*

QAM - *Quadrature amplitude modulation*

SBS - *Stimulated Brillouin scattering*

SER - *Symbol error rate*

SNR - *Signal-to-noise ratio*

SPM - *Self-phase modulation*

SRS - *Stimulated Raman scattering*

SSMF - *Standard single mode fiber*

XPM - *Cross-phase modulation*

WDM - *Wavelength-division Multiplex*

Sumário

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

1.1 Evolução das comunicações ópticas

A constante evolução dos sistemas de comunicações ópticas, que se deu início no começo da década de 1980, tem sido motivada pelo constante aumento na demanda por capacidade em sistemas de telecomunicações (AGRAWAL, 2016). O último grande salto de geração foi viabilizado por processadores digitais de sinais (DSP - *Digital signal processors*) de alto desempenho, que permitiram a utilização de formatos avançados de modulação, capazes de codificar informação em amplitude, fase e polarização, provendo grande aumento na eficiência espectral e possibilitando um incremento sem precedentes na capacidade de sistemas de comunicações ópticas modernos (KIKUCHI, 2015).

Historicamente, o desenvolvimento de tecnologias capazes de transmitir informação por fibras ópticas é algo relativamente recente. Os estudos preliminares que viabilizaram a implementação desta tecnologia na área das telecomunicações tiveram início na década de 1960 com a invenção e demonstração do *laser* para tais finalidades e desenvolvimento de fibras de baixa atenuação, que mais tarde viriam a ser utilizados em grande parte dos sistemas de comunicações ópticas modernos (AGRAWAL, 2012).

A evolução de estudos relacionados a materiais e processos utilizados na fabricação de dispositivos e fibras ópticas permitiu o desenvolvimento dos primeiros sistemas de comunicação por fibra óptica, que surgiram por volta de 1975 e tornaram-se comerciais em 1980 (SANFERRARE, 1987). Embora os sistemas de primeira geração operassem com taxas de 45 Mb/s e *spans* de até 10 km, a capacidade de enlaces ópticos foi aumentando significativamente com o passar dos anos (KOGELNIK, 2000), sendo impulsionada por fatores como o desenvolvimento de lasers capazes de operar em comprimentos de onda nos quais a fibra possui menores perdas (1,3 μm na segunda geração e 1,5 μm nas gerações posteriores) e a utilização de fibras monomodo, responsáveis por uma diminuição significativa na dispersão do canal (AGRAWAL, 2012).

Um dos saltos tecnológicos mais significativos em sistemas de comunicações ópticas ocorreu no início da quarta geração com a utilização de multiplexação por divisão de comprimento de onda (WDM – *Wavelength-Division Multiplexing*), técnica que possibilitou a implementação de sistemas com taxas superiores a 1 Tb/s. A WDM tornou-se viável após o desenvolvimento dos primeiros amplificadores a fibra dopada com érbio (EDFA –

Erbium Doped Fiber Amplifier), no final da década de 1980, que permitiram a amplificação simultânea de múltiplos canais no domínio óptico, operando na faixa de 1530 – 1570 nm e dispensando a necessidade de conversão optoeletrônica e amplificação de múltiplos canais no domínio elétrico em nós intermediários da rede (BECKER; OLSSON; SIMPSON, 1999).

Embora a capacidade de sistemas de comunicações ópticas tenha crescido ao longo dos anos, a utilização de múltiplos canais causou o aumento da potência total transmitida em enlaces ópticos. Desta forma, o estudo de efeitos não lineares sobre a capacidade de sistemas de ondas luminosas despertou grande interesse nos últimos anos (MITRA; STARK, 2001; MECOZZI; ESSIAMBRE, 2012). Tais estudos extendem o conceito de capacidade de canal, proposto por Shannon no contexto da teoria da informação (SHANNON, 1948), e demonstram que as não linearidades reduzem a eficiência espectral do sistema quando altas potências são transmitidas pela fibra (ESSIAMBRE *et al.*, 2010; MECOZZI; ESSIAMBRE, 2012).

Tem-se que, independentemente do comprimento do enlace, existe uma potência de transmissão ótima que determina, dados os parâmetros do sistema, o ponto ao qual o canal passa a ser limitado por efeitos não lineares (MECOZZI; ESSIAMBRE, 2012). Desta forma, a compensação desses efeitos pode resultar no deslocamento da potência de transmissão ótima, resultando no aumento das taxas de bit e/ou do comprimento máximo do enlace.

Além da compensação de adversidades, o monitoramento da qualidade do sinal por meio de algoritmos de baixa latência em sistemas ópticos modernos tem papel fundamental na identificação de falhas, viabilizando uma gestão preventiva mais eficiente que permita a configuração da rede antes da ocorrência de falhas e, consequentemente, interrupção total da comunicação. Dentre as principais métricas de qualidade em sistemas de comunicações ópticas coerentes digitais a serem monitoradas, podemos mencionar a razão erro de bit (BER - *bit error rate*), que pode ser estimada por meio da técnica de magnitude de vetor de erro (EVM - *error vector magnitude*), aplicável a sistemas limitados por ruído gaussiano (FATADIN, 2016; MEHRA; SADAWARTI; SINGH, 2017). Em contrapartida, efeitos não lineares geram distorções na constelação de sinais que dificultam a modelagem da função densidade de probabilidade dos diferentes símbolos, impossibilitando assim a aplicação do EVM para estimar a BER do sistema.

Com base neste contexto, o presente trabalho apresenta um método de estimativa de BER que faz uso de algoritmos de redes neurais artificiais (ANN – *Artificial neural network*),

especificamente redes neurais convolucionais (CNN - *Convolutional neural network*). As estimativas são realizadas por meio do processamento de diagramas de constelação de sinais para sistemas de comunicações ópticas digitais coerentes limitados ou não por efeitos não lineares.

1.2 Descrição do problema e contribuição do trabalho

Nos últimos anos, diversos autores vêm desenvolvendo estudos buscando compensar diferentes tipos de adversidades por meio da utilização de algoritmos baseados em inteligência artificial. Atualmente, existem trabalhos publicados na literatura que buscam compensar efeitos como ruído de fase (TORRES *et al.*, 2016), dispersão por modo de polarização (PMD – *Polarization mode dispersion*) (WU *et al.*, 2009) e não linearidades (JARAJREH *et al.*, 2014; ERIKSSON; BÜLOW; LEVEN, 2017; SILVA *et al.*, 2021).

Analogamente, a análise da qualidade de transmissão por meio de algoritmos de aprendizado de máquina (ML - *Machine learning*) tem apresentado bons resultados para a predição de diferentes figuras de mérito, tais como relação sinal ruído (SNR - *Signal-to-noise ratio*) e BER, porém dependem da utilização de blocos massivos de dados referentes à diversos parâmetros da rede, tais como a quantidade de enlaces e seus comprimentos, volume de tráfego e tipo de modulação (SAMADI *et al.*, 2017; BARLETTA *et al.*, 2017; ROTTONDI *et al.*, 2018).

Diferentemente das aplicações conhecidas na literatura, o presente trabalho implementa algoritmos de CNNs capazes de predizer a BER unicamente por meio do processamento de constelações de sinais. Estes algoritmos são capazes de generalizar esta estimativa para redes ópticas passivas (PON - *Passive optical networks*), independentemente de seu comprimento ou potência de transmissão, estando elas limitadas ou não por não linearidades.

1.3 Organização do documento

A continuação do presente trabalho é dividido como se segue:

O **Capítulo 2** apresenta uma descrição teórica do funcionamento de sistemas de comunicações ópticas digitais coerentes, dos principais impedimentos físicos que atuam

sobre este tipo de sistema e do EVM, tido como técnica convencional para estimar a BER em sistemas limitados por ruído gaussiano.

O **Capítulo 3** introduz uma descrição geral sobre redes neurais artificiais e redes neurais convolucionais, abordando assuntos relacionados à estrutura dos algoritmos, forma de funcionamento, procedimentos de treino, teste e validação cruzada.

O **Capítulo 4** descreve as especificações técnicas dos diferentes sistemas de comunicações ópticas considerados neste trabalho, bem como o detalhamento das informações topológicas das CNNs implementadas.

O **Capítulo 5** apresenta os resultados obtidos utilizando CNNs, comparando os desempenhos do EVM e destes algoritmos na estimativa de BER para os mais diferentes cenários, bem como a variação de desempenho considerando o processamento de histogramas de constelações com diferentes quantidades de símbolos e bins.

Por fim, o **Capítulo 6** destaca as principais conclusões obtidas ao longo do desenvolvimento da presente pesquisa.

2 Sistemas de comunicações ópticas

2.1 Sistemas de comunicações ópticas digitais coerentes

Embora ainda existam sistemas de comunicações ópticas baseados em detecção direta empregando modulação *On-Off keying* (OOK), os sistemas ópticos digitais coerentes passaram a se popularizar a partir dos anos 2010, principalmente por possibilitarem a utilização de modulações com maior eficiência espectral (AGRAWAL, 2012).

Sistemas de comunicações ópticas digitais coerentes modernos operam com demodulação intradina, que permite separar as componentes em fase e quadratura do sinal transmitido após a conversão ao domínio elétrico. Inicialmente, estes sistemas foram desenvolvidos para enlaces de longa distância, mas passaram a ser progressivamente utilizados em enlaces menores, como PONs (*Passive optical network*) (LAVERY *et al.*, 2010). Como exemplo, a Figura 1 apresenta um diagrama de blocos do sistema adotado no presente trabalho, que constitui uma rede óptica passiva coerente de longa distância (*Long reach passive optical network*, LR-PON) operando em dupla polarização.

É possível notar no diagrama de blocos (Figura 1) que a informação a ser transmitida é direcionada a conversores serial-paralelo, responsáveis por ordenar sequencialmente os bits. Os conversores são então conectados a mapeadores 16-QAM (*Quadrature amplitude modulation*), cuja finalidade é mapear os bits nas constelações desejadas (uma para cada polarização), seguidos por filros de Nyquist com fator de roll-off de 10% e por conversores digitais-analógicos, que geram os sinais modulantes das componentes em fase e quadratura. A modulação óptica se dá por meio de moduladores Mach-Zenhder duplo-paralelos (*Mach-Zenhder modulator*, MZM), conectados à um *laser* de diodo (LD_1) que, por meio de um divisor de feixes polarizados, geram sinais ópticos com polarizações ortogonais. Ainda no transmissor, combinam-se os sinais modulados e utiliza-se um amplificador EDFA para ajustar a potência do sinal transmitido. Na Figura 1, o canal óptico é composto por um trecho de fibra monomodo padrão (*Standard single mode fiber*, SSMF).

Por fim, ao se analisar o receptor do sistema, nota-se que os sinais ópticos com polarização ortogonal são divididos e combinados com dois sinais derivados de um *laser* de diodo local (LD_2), que opera próximo à frequência nominal da portadora. Este procedimento é realizado por meio de redes híbridas de 90^{o}, de modo que nas saídas o oscilador local é defasado em 0^{o}, 90^{o}, 180^{o} e 270^{o}.

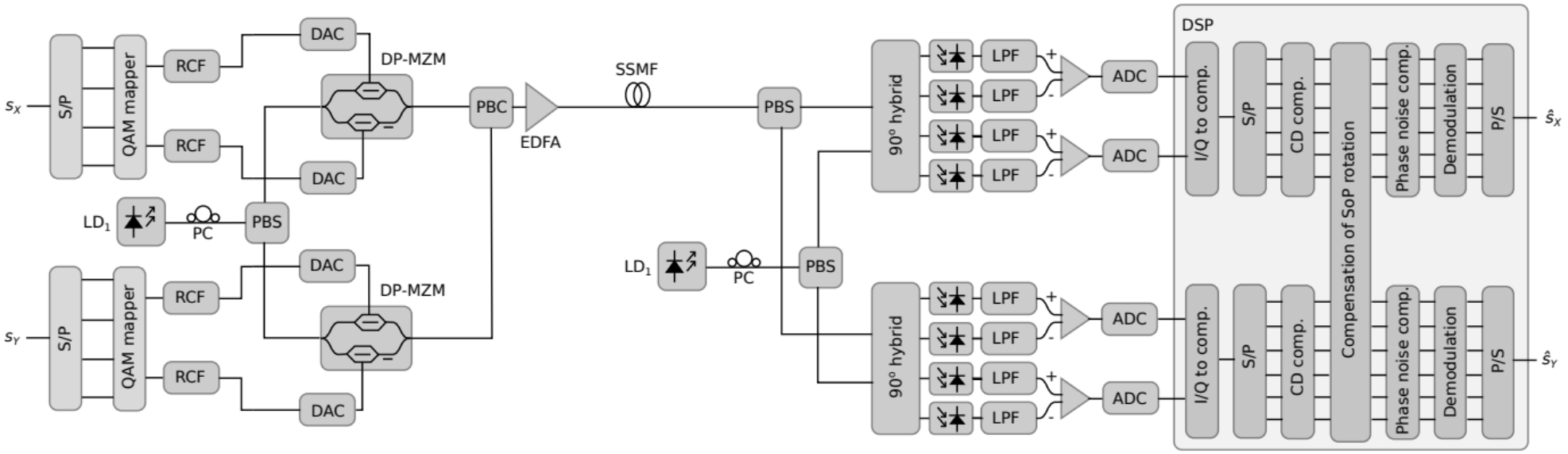


Figura 1 – Sistema LR-PON coerente DP-16QAM simulado. S/P: Conversão serial-paralela (*serial-to-parallel conversion*). DAC: Conversor digital analógico (*digital-to-analog converter*). LD: Laser de diodo (*laser diode*). PBS: Divisor de feixes polarizados (*polarizing beam splitter*). DP-MZM: Modulador Mach-Zenhder duplo em paralelo (*dual parallel-Mach-Zenhder modulator*). PBC: Combinador de feixes polarizados (*polarizing beam combiner*). EDFA: Amplificador de fibra dopada a érbio (*erbium-doped fiber amplifier*). VOA: Atenuador óptico variável (*variable optical attenuator*). SSMF: Fibra monomodo padrão (*standard single mode fiber*). Att: Atenuador (*attenuator*). LPF: Filtro passa baixas (*low-pass filter*). ADC: Conversor analógico digital (*analog-to-digital converter*). DSP: Processamento digital de sinal (*digital signal processing*).

Posteriormente, as saídas de cada polarização passam por pares de fotodetectores balanceados limitados em banda, sendo esta limitação representada por filtros passa baixas (*Low-pass filter*, LPF) com frequência de corte igual a 75% da banda do sinal recebido. Após a fotodetecção, os sinais são diferencialmente amplificados, possibilitando a separação das componentes em fase e quadratura.

Posteriormente à digitalização do sinal mediante conversores analógico digitais (*Analog-to-digital converter*, ADC), as componentes em fase e quadratura são submetidas ao bloco de processamento digital de sinal (*Digital signal processing*, DSP). Este bloco é dividido em subsistemas responsáveis pela compensação de adversidades, sendo elas a dispersão cromática, rotação por estado de polarização e compensação de ruído de fase linear.

2.2 Impedimentos em sistemas de comunicações ópticas

De modo geral, sistemas de comunicações são suscetíveis a adversidades que degradam a qualidade do sinal transmitido, aumentando a probabilidade de ocorrência de erros na detecção da informação no receptor. Em particular, canais ópticos são afetados por diversos tipos de impedimentos, que por sua vez estão relacionados a efeitos lineares e não lineares (AGRAWAL, 2012).

As subseções a seguir apresentarão uma breve descrição dos principais impedimentos lineares e não lineares que afetam um sistema de comunicação óptica implementado com fibra SSMF.

2.2.1 Impedimentos lineares

Os principais efeitos lineares que afetam sistemas de comunicações ópticas estão relacionados a atenuação, dispersão e ruído (AGRAWAL, 2012). Considerando um enlace implementado com fibras monomodo, podemos observar as seguintes adversidades:

- **Dispersão cromática:** também conhecida como dispersão de velocidade de grupo (GVD - *Group-velocity dispersion*), este efeito está relacionado com a velocidade de grupo associada ao modo de propagação fundamental da fibra, que varia em função da frequência. Desta forma, as diferentes componentes espectrais do sinal

se propagam com velocidades diferentes, ocasionando um alargamento de pulso e consequentemente interferência intersimbólica (AGRAWAL, 2012). Este efeito pode ser compensado por meio da utilização de fibras compensadoras de dispersão (DCF - *Dispersion-compensating fiber*) ou por meio de processamento digital de sinais. O sistema implementado no presente trabalho faz uso de compensação de GVD por meio de DSP.

- **Dispersão do modo de polarização:** A PMD é um efeito que está relacionado aos pequenos desvios da perfeita simetria cilíndrica, que levam à birrefringência [1]. Considerando birrefringência constante, a elipsidade do núcleo quebra a degenerescência das duas polarizações do modo fundamental da fibra, ocasionando velocidades de grupo distintas para cada polarização, resultando também em interferência intersimbólica. Devido à natureza estocástica da PMD, modelos analíticos que descrevem este efeito são bastante complexos (AGRAWAL, 2012). Técnicas de DSP conseguem compensar a contribuição linear deste efeito.
- **Atenuação:** A atenuação em fibras ópticas está relacionada com os processos de absorção e espalhamento de luz no material. O mecanismo fundamental de perda em fibras modernas que operam no comprimento de onda de 1550 nm é conhecido como espalhamento Rayleigh, e ocorre devido à existência de flutuações microscópicas de densidade em uma escala menor que o comprimento de onda do sinal óptico (λ) (AGRAWAL, 2012). Este fenômeno é um efeito elástico, uma vez que os fótons mantém sua frequência original após o processo de espalhamento (AGRAWAL, 2000). A atenuação também pode ocorrer em outros dispositivos passivos da rede de distribuição, normalmente devido à perdas de inserção e derivação. A compensação de atenuação se dá por meio da utilização de amplificadores, que podem atuar de forma localizada ou distribuída.
- **Ruído de fase:** O ruído de fase é particularmente crítico em sistemas coerentes (TORRES *et al.*, 2016), e está relacionado com a ocorrência de emissão espontânea nos lasers do transmissor e receptor, que causam variações de fase no campo gerado. O ruído de fase é um processo estocástico não ergódico, que gera uma rotação da constelação, e está diretamente relacionado com a largura de linha dos *lasers*. Deste

[1] Condição em que um mesmo meio (fibra) apresenta dois índices de refração diferentes. Em fibra ópticas, a birrefringência está associada ao desvio da perfeita simetria cilíndrica, uma vez que os índices nas direções dos semieixos maior e menor são diferentes (AGRAWAL, 2012)

modo, quanto menor é este parâmetro, maior o tempo de coerência da fase (BRITO *et al.*, 2015). O sistema implementado no presente trabalho faz uso de compensação de ruído de fase por meio de DSP.

- **Ruído aditivo:** Em sistemas de comunicações ópticas, o ruído aditivo pode ocorrer no domínio óptico ou elétrico. No domínio óptico, a principal fonte de ruído é denominada emissão espontânea amplificada (ASE - *amplified spontaneous emission*) e está relacionada com a ocorrência de emissões espontâneas nos amplificadores, gerando fótons com amplitude e fase aleatórias que podem ser amplificados conjuntamente ao sinal (BECKER; OLSSON; SIMPSON, 1999). Devido a ausência de amplificadores na rede de distribuíção, o ruído ASE pode ser desconsiderado em sistemas ópticos passivos. No domínio elétrico, o ruído ocorre durante a fotodetecção, podendo ser classificado como térmico, causado pelo movimento aleatório dos elétrons, e *shot*, relacionado à natureza discreta da corrente elétrica (elétrons são quantizados). A densidade espectral de potência do ruído *shot* é diretamente proporcional à corrente gerada no fotodetector (AGRAWAL, 2012). Sendo assim, esse tipo de ruído é significativo em sistemas ópticos digitais coerentes, uma vez que o receptor mistura o sinal à um oscilador local com um nível de potência consideravelmente alto.

2.2.2 Impedimentos não lineares

Em sistemas de comunicações ópticas, as não linearidades dominantes são geradas pelas fibras, impactando diretamente na capacidade destes sistemas. Estes efeitos estão diretamente relacionados com a polarização do meio dielétrico (sílica) devido a propagação de campos eletromagnéticos intensos (AGRAWAL, 2012). Os principais impedimentos não lineares são:

- **Espalhamento estimulado de luz:** diferentemente do espalhamento Rayleigh (explicado na Subseção 2.2.1), espalhamentos estimulados de luz são fenômenos inelásticos. No caso de Stokes, a frequência da luz espalhada é menor que a do sinal que a originou. Neste tipo de efeito, um fóton é aniquilado gerando um novo fóton e um fônon, ambos com energia menor à do fóton que os originou (AGRAWAL, 2000). Em fibras ópticas existem dois tipos de espalhamento estimulado de luz, denominados espalhamento estimulado de Brillouin (SBS - *stimulated Brillouin scattering*) e

espalhamento estimulado de Raman (SRS - *stimulated Raman scattering*), e ocorrem somente quando a potência do sinal é superior a seus respectivos limiares. O limiar de Brillouin é consideravelmente menor que o de Raman, porém, os efeitos de ambos são desprezíveis para os sistemas simulados no presente trabalho, uma vez que a modulação utilizada possui portadora suprimida (AGRAWAL, 2012; AGRAWAL, 2000).

- **Efeito Kerr:** este efeito está relacionado com a modulação de fase não linear em sistemas de comunicações ópticas (AGRAWAL, 2000), e é o impedimento não linear mais considerável nos sistemas apresentados neste trabalho. Desta forma, este fenômeno será descrito com maiores detalhes na Subseção 2.2.2.

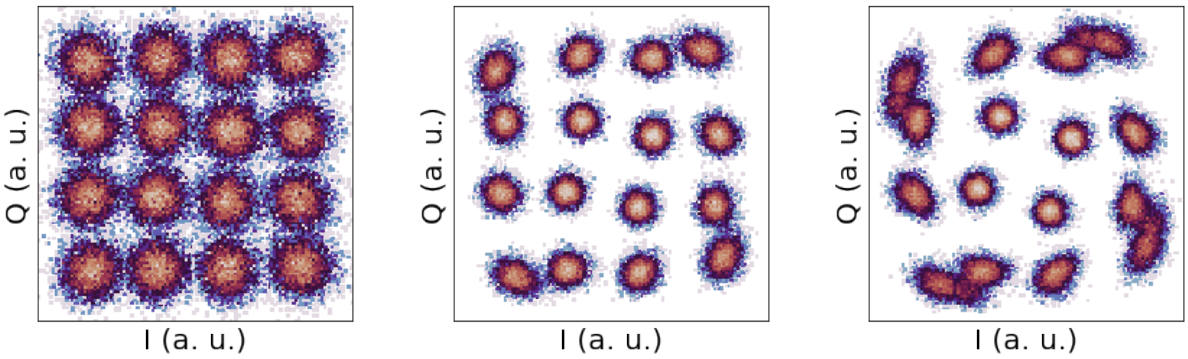
Modulação de fase não linear devido a efeito Kerr

Os sistemas de comunicações ópticas digitais coerentes sofrem modulação de fase não linear devido ao efeito Kerr (em particular o efeito Kerr óptico) (AGRAWAL, 2012). Esta modulação está relacionada ao comportamento não linear do índice de refração do material, que varia linearmente de acordo com a intensidade do sinal óptico aplicado (AGRAWAL, 2000). Para descrever esta variação, podem-se modelar os índices de refração do núcleo (n_1') e da casca (n_2') da fibra de acordo com a seguinte equação:

$$n_j' = n_j + I\bar{n}_2, \qquad j = 1, 2 \tag{1}$$

sendo I a intensidade óptica localizada e $\bar{n}_2$ o coeficiente de índice não linear, que é da ordem de $2,6 \times 10^{-20}\ \ m^2/W$ para sílica fundida e pode variar de acordo com a dopagem do núcleo da fibra. Embora $\bar{n}_2 \ll n_1, n_2$', a modulação não linear de fase torna-se significativa para enlaces ópticos longos e/ou altos valores de potência de transmissão (AGRAWAL, 2000).

O efeito Kerr causa três fenômenos distintos: automodulação de fase (*SPM - Self-phase modulation*), modulação de fase cruzada (*XPM - Cross-phase modulation*) e mistura de quatro ondas (*FWM - Four wave mixing*). De modo geral, modulação de fase cruzada e mistura de quatro ondas são efeitos que ocorrem exclusivamente quando se faz uso de



(a) (b) (c)

Figura 2 – Constelações recebidas (16-QAM) em sistema óptico passivo *single channel* de 100 km de comprimento para potências transmitidas de (a) 3 mW, (b) 8 mW e (c) 13 mW. Valores de amplitude em unidades arbitrárias.

múltiplos canais, como é o caso de sistemas WDM, enquanto que a automodulação de fase ocorre em qualquer tipo de sistema de comunicação óptica.

Automodulação de fase não linear

Devido à existência de $\bar{n}_2$, a constante de fase do sinal óptico sofre alterações que geram uma fase não linear. Considerando um enlace com baixa dispersão e comprimento L, a fase não linear gerada pode ser descrita de acordo com a equação abaixo:

$$\Phi_{NL} = \int_0^L k_0 \left(\frac{P(z)}{A_{eff}}\right) \bar{n}_2 \, dz = \int_0^L \left(\frac{2\pi}{\lambda}\frac{\bar{n}_2}{A_{eff}}\right) P dz = \int_0^L \gamma P(z) dz = \gamma P_{in} L_{eff} \qquad (2)$$

sendo $P(Z)$ a potência óptica ao longo da fibra, A_{eff} a área modal efetiva, γ é o parâmetro não linear, P_{in} a potência óptica instantânea na entrada da fibra e L_{eff} o comprimento efetivo da fibra. Devido à variação de P_{in} com relação ao tempo, Φ_{NL} também apresenta dependência temporal. Desta forma, a alteração da fase não linear em função de P_{in} gera uma modulação autoinduzida, denominada automodulação de fase (AGRAWAL, 2000).

Um fato importante a se destacar é que caso a constelação transmitida possua símbolos com diferentes valores de energia, Φ_{NL} não será uniforme, uma vez que símbolos com energia mais elevada possuem maiores valores de amplitude e sofrem maiores rotações. Além disso, a distorção da constelação gerada pela automodulação de fase tende a ser mais significativa para potências de transmissão (LOP - *Launch optical power*) mais elevadas.

Como exemplo, a Figura 2 mostra as constelações recuperadas por meio de detecção intradina em um sistema de comunicação óptica digital coerente de um único canal e uma polarização, com baixa dispersão e comprimento de enlace de 100 km considerando diferentes valores de LOP. Nota-se que o efeito Kerr é pouco significativo quando a potência transmitida é de 3 mW, e que neste caso o sistema é limitado por ruído gaussiano. Considerando uma potência de transmissão de 8 mW, a automodulação de fase passa a ser perceptível, e tende a distorcer significativamente símbolos com maiores amplitudes.Para 13 mW de transmissão, a constelação é bastante distorcida, ocasionando um aumento significativo nas taxas de erro de símbolo (SER - *Symbol error rate*). Para este caso, faz-se necessária a utilização de algoritmos de equalização e/ou de detecção com regiões de decisão otimizadas, uma vez que os símbolos são dispostos de maneira não uniforme.

Modulação de fase cruzada

Analogamente ao que acontece na automodulação de fase, a relação linear entre índice de refração da fibra e intensidade de sinal óptico pode ocasionar outro efeito não-linear, conhecido como modulação de fase cruzada (AGRAWAL, 2012). Este tipo de fenômeno ocorre quando múltiplos canais trafegam em uma mesma fibra utilizando a técnica WDM. Sabe-se que, assim como descrito anteriormente, um canal individual gera variações de índice de refração na fibra que induzem a uma modulação de fase não linear (SPM). Quando mais de um canal óptico trafega neste enlace, estas variações de índice de refração ao longo do tempo afetam simultaneamente a todos, uma vez que o mesmo meio físico (fibra óptica) é compartilhado. O deslocamento de fase para o j-ésimo canal é descrito pela seguinte equação:

$$\Phi_{NL}^{j} = \gamma L_{eff} \left(P_j + 2 \sum_{m \neq j} P_m \right) \tag{3}$$

em que o somatório abrange os valores de potência de todos os canais do sistema. O fator 2 indica que o XPM é duas vezes mais eficaz que o SPM e tem origem na forma da suscetibilidade não linear (AGRAWAL, 2000).

Mistura de quatro ondas

A terceira forma com a qual o efeito Kerr óptico se apresenta em sistemas de comunicações ópticas é conhecida como mistura de quatro ondas (FWM). Este fenômeno se origina quando três sinais ópticos com frequências portadoras ω_1, ω_2 e ω_3 trafegam por um mesmo meio, gerando campo em frequências $\omega_4 = \omega_1 \pm \omega_2 \pm \omega_3$ (AGRAWAL, 2012). Embora a maioria das das componentes espectrais geradas não prosperem, uma vez que dependem do casamento de fase, algumas podem ter efeitos degradantes em sistemas WDM, especificamente quando as interferências são geradas em freqências próximas às de canais operantes, ocasionando *crosstalk* (AGRAWAL, 2000).

2.3 Estimativa de probabilidade de erro de símbolo por meio da magnitude do vetor de erro

De modo geral, o EVM é uma técnica que permite estimar a BER em sistemas de comunicações limitados por ruído branco gaussiano aditivo (AWGN - *Additive white Gaussian noise*), podendo ser aplicado em configurações que fazem uso dos mais diversos tipos de modulação (SHAFIK; RAHMAN; ISLAM, 2006).

Em formatos de modulação avançados, como é o caso do QAM M-ário, a informação é codificada em um sinal com amplitude e fase, podendo ser representada no plano complexo na forma de uma constelação de sinais. O EVM é definido como sendo a raíz quadrada do erro quadrático médio (RMS - *Root mean square*) de uma coleção de símbolos, sendo o erro representado pela distância euclidiana entre símbolos amostrados e os respectivos símbolos detectados (idealmente iguais aos transmitidos). A Figura 3 mostra o vetor de erro para um símbolo de uma constelação 16 QAM, apresentando o desvio do vetor de sinal recebido ($E_{r,i}$) com relação ao transmitido ($E_{t,i}$) por meio do vetor de erro ($E_{err,i}$) (SCHMOGROW *et al.*, 2011; FATADIN, 2016).

Considerando uma coleção formada por N símbolos recebidos e potência média de transmissão dada por P_a, representa-se o EVM de acordo com a seguinte expressão:

$$EVM_{rms} = \sqrt{\frac{\frac{1}{N}\sum_{i=1}^{N} |E_{r,i} - E_{t,i}|^2}{P_a}}. \tag{4}$$

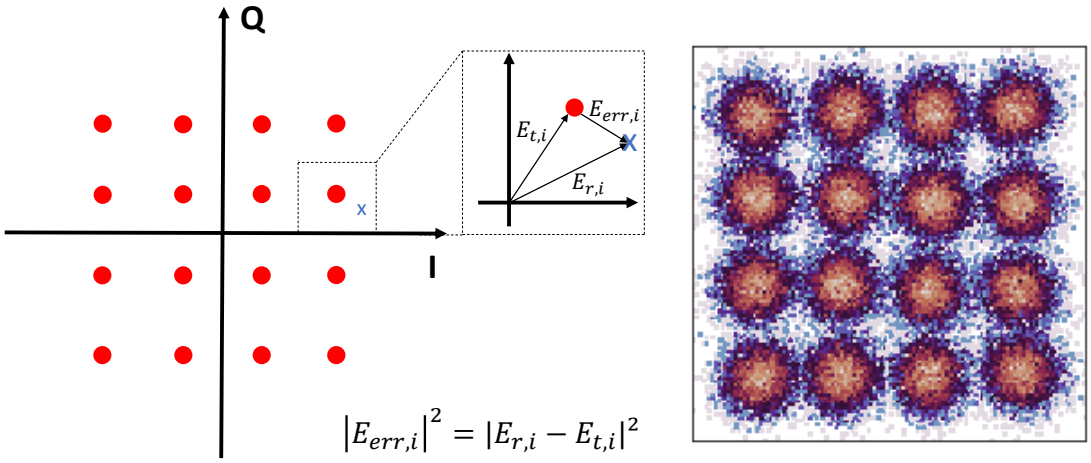


Figura 3 – Diagrama de constelação com vetor de erro expresso para sinal 16QAM e constelação sob efeito de ruído gaussiano aditivo.

O EVM se relaciona diretamente com a SNR do sinal de acordo com a seguinte expressão:

$$SNR \approx \frac{1}{EVM^2} = \frac{E_s}{N_0} \tag{5}$$

sendo E_s a energia do símbolo e $N_0/2$ a densidade espectral de potência do ruído gaussiano.

Em sistemas limitados por ruído, sabe-se que a BER é um efeito diretamente relacionado ao AWGN, que por sua vez pode ser representado em função do EVM (LATHI, 1998). Considerando constelações QAM M-árias, a expressão que relaciona a BER ao EVM é a seguinte (FATADIN, 2016; SCHMOGROW *et al.*, 2011):

$$BER = \frac{(1 - M^{-1/2})}{\frac{1}{2}\ log_2 M} erfc\left[\sqrt{\frac{3/2}{(M-1)EVM_{rms}^2}}\right] \tag{6}$$

em que M representa a quantidade de símbolos na constelação e $erfc$ é a função de erro complementar.

3 Redes neurais artificiais

Neste trabalho foram aplicados algoritmos baseados em CNNs para estimar a BER em sistemas de comunicações ópticas digitais coerentes por meio do processamento de imagens de constelações de sinais. Esses tipos de algoritmos fazem parte de uma classe específica de ANNs, e serão descritos com maiores detalhes neste capítulo.

ANNs, normalmente abreviadas "redes neurais", são algoritmos computacionais que se inspiram no funcionamento do sistema nervoso central de seres vivos, sendo capazes de adquirir e manter o conhecimento e torná-lo disponível para uso. Sabe-se que o cérebro é um sistema altamente complexo, não linear e paralelo, que possui alta capacidade de organização de seus constituintes (denominados neurônios) para realizar atividades diversas, como reconhecimento de padrões, controle motor e percepção (HAYKIN, 2007).

Desta forma, redes neurais fazem uso da interligação de células computacionais simples, denominadas "neurônios artificiais", para realizar uma tarefa ou função de interesse. Em uma de suas descrições mais generalistas, este tipo de sistema é descrito como um processador paralelamente distribuído que possui uma propensão ao armazenamento de conhecimento experimental. Este conhecimento deve ser adquirido externamente por meio do processo de aprendizagem, que é responsável por criar conexões neurais, normalmente denominados como pesos sinápticos, que armazenam o conhecimento (ALEKSANDER; MORTON, 1990). Na maioria dos casos, as ANNs são algoritmos de aprendizado de máquina que se baseiam no paradigma de aprendizagem supervisionado. Na seção a seguir, será apresentada uma breve descrição do modelo de funcionamento de neurônios artificiais.

3.1 Neurônios artificiais

De forma geral, neurônios artificiais são unidades simples não lineares de processamento. O diagrama da Figura 4 apresenta um modelo de um neurônio artificial conhecido como Perceptron, responsável por compor a estrutura dos mais diferentes tipos de arquiteturas de redes neurais.

Estas unidades processadoras recebem sinais de entrada, representados pelo conjunto $\{x_1, x_2, x_3, ..., x_n\}$, e os ponderam por meio dos elos de conexão, definidos como conjunto

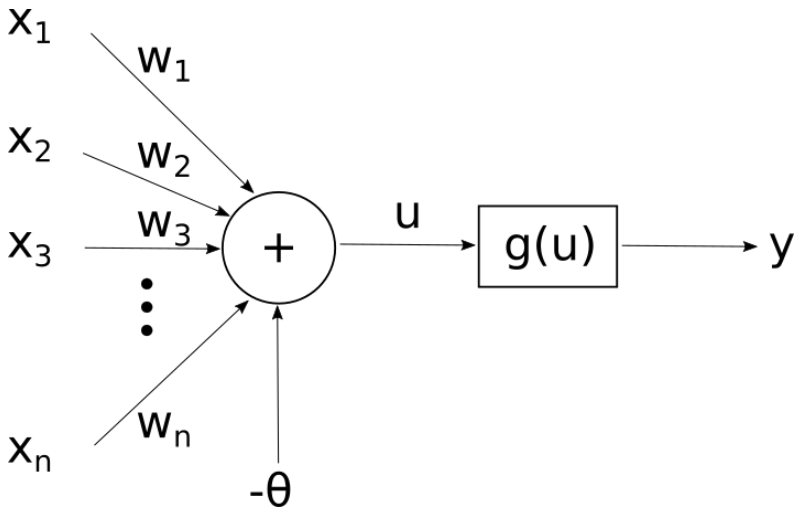


Figura 4 – Modelo de neurônio artificial perceptron.

de pesos sinápticos $\{\omega_1, \omega_2, \omega_3, ..., \omega_n\}$. É válido ressaltar que estes pesos podem assumir valores tanto positivos como negativos.

Após multiplicar cada uma das entradas x_i pelo seu respectivo peso ω_i, estes sinais são agregados e produzem uma combinação linear das entradas, comumente denotada por potencial de ativação (u). Ressalta-se que u também considera a soma ou subtração de um valor constante θ, denominado na literatura como *bias* (ALPAYDIN, 2009).

Por fim, o neurônio artificial aplica uma função de ativação a u com a finalidade de combinar as entradas ponderadas, tipicamente de forma não linear. Partindo do procedimento previamente descrito, pode-se descrever um neurônio artificial por meio do seguinte par de equações (ALPAYDIN, 2009):

$$u = \sum_{i=1}^{n} \omega_i \cdot x_i - \theta \tag{7}$$

e

$$y = g(u) \tag{8}$$

A função de ativação, apresentada na Figura 4 como $g(u)$, descreve a saída de um neurônio em função de u e pode variar de acordo com a posição em que ele se encontra dentro da estrutura da ANN ou com o tipo de tarefa a ser executada (SHARMA; SHARMA, 2017). Embora existam e sejam aplicadas diversas funções para esta finalidade, serão subsequentemente apresentadas e descritas as funções mais relevantes para a implementação do trabalho proposto.

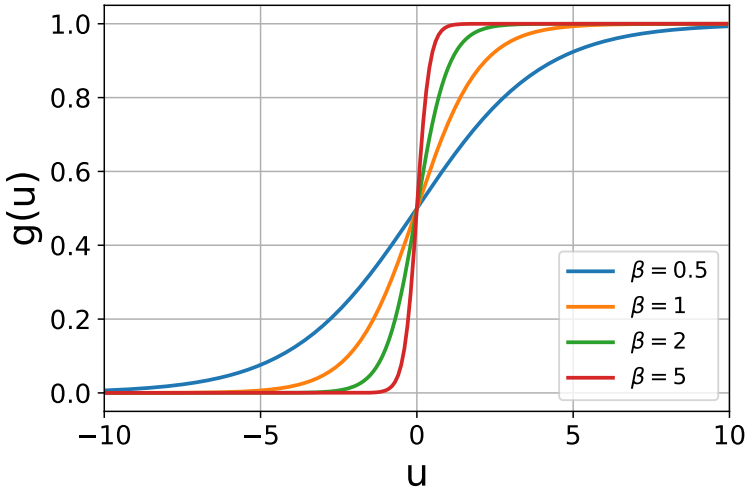


Figura 5 – Função logística (sigmoide) considerando diferentes valores de β.

3.1.1 Função sigmoide

A função sigmoide, muitas vezes apresentada na literatura como função logística, é um dos tipos de função de ativação mais utilizados na construção de ANNs (HAYKIN, 2007). Trata-se de uma função estritamente crescente que assume valores de saída no intervalo de [0, 1] e é descrita pela seguinte equação:

$$g(u) = \frac{1}{1 + e^{-\beta \cdot u}}, \tag{9}$$

na qual β é o *parâmetro de inclinação* e se relaciona com a inclinação da função e seu ponto de inflexão.

A função sigmoide é representada na Figura 5 considerando diferentes valores de β. Tem-se que ao adotar valores altos para β, a inclinação no ponto de inflexão $(0, g(0))$ tende a ser cada vez maior, de modo que, conforme $\beta \to \infty$, $g(u)$ tende à função degrau unitário, também conhecida como função de Heaviside (ALPAYDIN, 2009).

Uma característica importante da função sigmoide é o fato de que ela é diferenciável, sendo este um fator extremamente desejável na teoria de redes neurais, assim como será tratado posteriormente na Subseção 3.2.1.

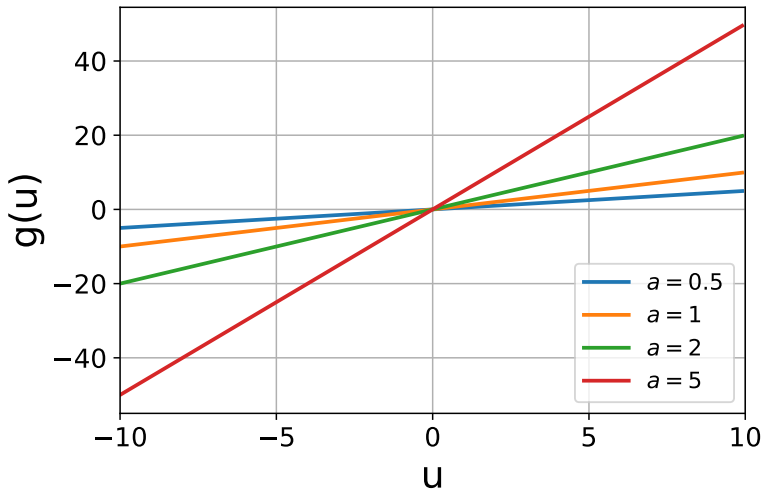


Figura 6 – Função linear considerando diferentes valores de a.

3.1.2 Função linear

A função de ativação linear é descrita por uma reta, sendo a saída diretamente proporcional ao parâmetro de entrada (u) (SHARMA; SHARMA, 2017). Além de ser diferenciável, o gradiente da função é diferente de zero, constante e independente de u para $u \in \mathbb{R}$. Pode-se descrever uma função de ativação linear pela seguinte equação:

$$g(u) = au, \tag{10}$$

em que a é a inclinação da reta.

A função linear é representada na Figura 6 considerando diferentes valores de a. Ela é normalmente utilizada para casos em que a saída de um neurônio artificial não é limitada a intervalos específicos, como camadas de saídas em ANNs de regressão (abordadas na Subseção 3.2.1). Quando é unicamente utilizada, inviabiliza a detecção de padrões complexos nos dados.

3.1.3 Função ReLU

A função de unidade linear retificada (*rectified linear unit*), denotada ReLU, tem sido amplamente utilizada por apresentar bom desempenho em diversas aplicações envolvendo ANNs, incluindo arquiteturas específicas para processamento de imagem (XU *et al.*, 2015).

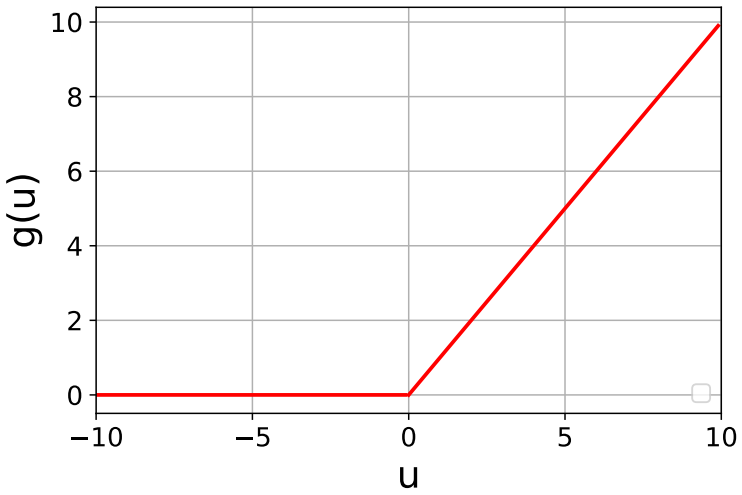


Figura 7 – Função RelU.

Ela é definida pelo valor máximo entre 0 e a entrada em questão, podendo ser descrita pela seguinte equação:

$$g(u) = ReLU(u) = max(0, u) = \begin{cases} 0 \ se \ u \leq 0 \\ u \ se \ u > 0. \end{cases} \tag{11}$$

Embora possa ser aplicada para outros casos, este tipo de função de ativação tem se mostrado particularmente relevante principalmente para CNNs, pois é capaz de suprimir valores negativos (irrepresentáveis na forma de *pixels*). Embora não seja diferenciável, esta função de ativação pode afetar positivamente o processo de aprendizagem, uma vez que ela não sofre saturação para valores altos de u, ao contrário do que ocorre em funções como a sigmoide (XU *et al.*, 2015). A função ReLU pode ser observada na Figura 6.

3.2 Arquiteturas de redes neurais

Define-se por arquitetura a forma com a qual os neurônios artificiais estão conectados na rede, estando este fator intimamente ligado ao algoritmo de treinamento (HAYKIN, 2007). Em particular para este trabalho, mostram-se importantes duas arquiteturas de rede: redes neurais perceptron multicamadas e CNNs. Essas arquiteturas serão descritas nas seções subsequentes.

3.2.1 Redes neurais perceptron multicamadas

Este tipo de arquitetura de rede neural artificial, normalmente conhecida como redes *feedforward* (alimentadas adiante) ou perceptron multicamadas (MLP - *Multilayer Perceptron*), é caracterizada pela existência de um fluxo unidirecional (com exceção da etapa de treinamento) da informação na rede, que ocorre da entrada para a saída. A organização de neurônios artificiais nesta arquitetura de rede se dá por três tipos distintos de camadas (SILVA; SPATTI, 2010), denominadas:

- **Camada de entrada:** Se projeta nas camadas ocultas, mas não vice-versa. Recebe as informações de entrada do sistema, normalmente de forma normalizada. A quantidade de neurônios de entrada depende da quantidade de parâmetros a serem considerados no problema (ALPAYDIN, 2009);
- **Camadas intermediárias (escondidas ou ocultas):** Sua função é intervir de forma útil entre a camada de entrada e a saída. Agregam conexões sinápticas à rede que podem melhorar a habilidade de extrair estatísticas de ordem elevada da tarefa a ser realizada (CHURCHLAND; SEJNOWSKI, 1994). São responsáveis por grande parte do processamento interno da rede.
- **Camada de saída:** Além de realizar parte do processamento interno, a camada de saída também é responsável por constituir a resposta global da rede de acordo com a tarefa a ser realizada (ALPAYDIN, 2009). Constitui a última camada de neurônios, e é indispensável para redes neurais de modo geral.

A Figura 8 apresenta uma rede neural perceptron multicamada com duas camadas escondidas considerando n entradas e m saídas. É válido destacar que o modelo representa uma rede totalmente conectada, também conhecida como rede densa, pois neste caso os nós de uma determinada camada se conectam a todos os nós da camada adjacente no sentido de propagação da informação. Caso haja ausência de determinadas conexões sinápticas, a rede pode então ser classificada como rede parcialmente conectada (ALPAYDIN, 2009).

Descrição matemática de camadas do tipo *feedforward*

Para descrever o funcionamento de uma ANN *feedforward*, podemos assumir uma rede com a topologia apresentada na Figura 8. Nesta arquitetura, cada neurônio artificial

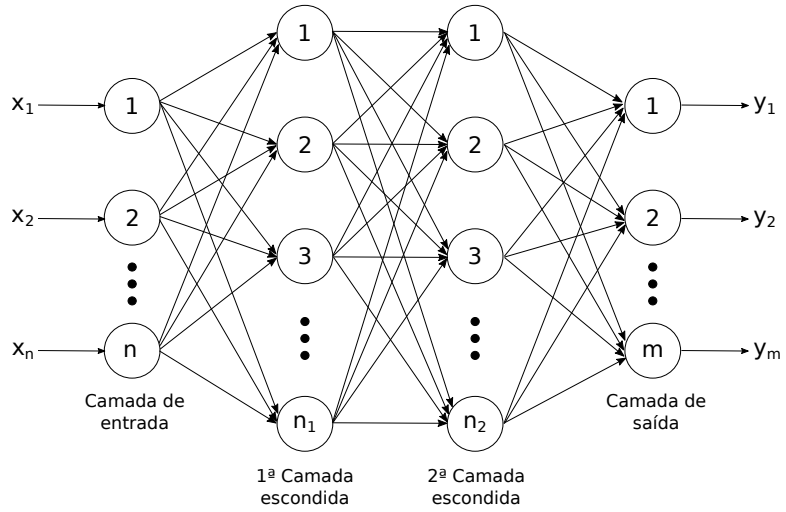


Figura 8 – Exemplo de uma rede neural artificial *feedforward* com n entradas, m saídas e duas camadas escondidas.

que compõe as camadas escondidas ou de saída é alimentado pelas saídas dos neurônios da camada anterior, seguindo o procedimento descrito previamente. Como exemplo, a equação abaixo apresenta as saídas de cada neurônio considerando uma camada escondida de comprimento j:

$$\begin{aligned} a_1 &= g\left(\omega_{01}x_0 + \omega_{11}x_1 + \omega_{21}x_2 + ... + \omega_{n1}x_n\right) \\ a_2 &= g\left(\omega_{02}x_0 + \omega_{12}x_1 + \omega_{22}x_2 + ... + \omega_{n2}x_n\right) \\ a_3 &= g\left(\omega_{03}x_0 + \omega_{13}x_1 + \omega_{23}x_2 + ... + \omega_{n3}x_n\right) \\ &\vdots \\ a_j &= g\left(\omega_{0j}x_0 + \omega_{1j}x_1 + \omega_{2j}x_2 + ... + \omega_{nj}x_n\right), \end{aligned} \tag{12}$$

sendo $x_0 = 1$, $\omega_0 = \theta$ e $g(\cdot)$ a função de ativação adotada para os neurônios artificiais desta determinada camada.

As entradas e saídas de cada uma das camadas que compõe uma ANN podem ser representadas e processadas na forma vetorial com a finalidade de facilitar a implementação computacional deste tipo de algoritmo, assim como possibilitar a utilização de uma nomenclatura mais compacta (ALPAYDIN, 2009; HAYKIN, 2007). Para isso, é preciso inicialmente descrever as entradas e saídas de cada camada como vetores colunas, de modo que o vetor saída seja resultante de uma operação entre o vetor de entrada e uma matriz composta por pesos sinápticos.

Considerando uma camada com n entradas e m neurônios artificiais, é possível descrever os vetores de entrada (X) e saída (Y), bem como a matriz de pesos (Ω) de acordo com:

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \Omega = \begin{bmatrix} \omega_{01} & \omega_{02} & \omega_{03} & ... & \omega_{0m} \\ \omega_{11} & \omega_{12} & \omega_{13} & ... & \omega_{1m} \\ \omega_{21} & \omega_{22} & \omega_{23} & ... & \omega_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega_{n1} & \omega_{n2} & \omega_{n3} & ... & \omega_{nm} \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix} \tag{13}$$

Na Equação 13, cada coluna da matriz Ω representa os pesos sinápticos de um único neurônio para cada uma das n entradas, ao passo que cada linha corresponde aos pesos de cada neurônio para uma entrada específica. Sendo assim, a expressão que relaciona os vetores de entrada e saída com a matriz Ω é descrita por:

$$Y = g(U) = g(\Omega^T X) \tag{14}$$

em que U, denominado vetor potencial de ativação, é dado por:

$$U = \Omega^T X. \tag{15}$$

Considerando o fluxo natural de uma rede perceptron multicamadas, as operações das camadas posteriores repetem este mesmo procedimento, assumindo que a entrada da i-ésima camada seja tal que $X_i = Y_{i-1}$. A propagação da informação pela rede partindo da camada de entrada até a saída é denominada *forward propagation* (ALPAYDIN, 2009).

Técnicas de treinamento para redes neurais artificiais *feedforward*

Assim como já mencionado, ANNs são algoritmos de ML que geralmente se baseiam no paradigma de aprendizagem supervisionada, portanto necessitam de uma etapa de

treinamento para gerar e armazenar conhecimento experimental por meio da otimização de pesos sinápticos (SILVA; SPATTI, 2010). Este procedimento é realizado por meio da minimização de uma função de custo médio $J(\Omega)$ idealmente contínua e diferenciável, capaz de mapear os pesos Ω e associá-los de alguma forma à saída esperada (HAYKIN, 2007).

O treinamento de ANNs perceptrons multicamadas faz uso do algoritmo de *back-propagation*, normalmente aplicado sucessivamente em duas etapas específicas (HAYKIN, 2007; ALPAYDIN, 2009). A primeira delas é denominada *forward propagation* (apresentada na Seção 3.2.1), e consiste na inserção das amostras de um bloco de teste (*training set*) na rede e na propagação da informação até a saída. Durante esta etapa, obtém-se as respostas da ANN mantendo-se inalterados os pesos e *bias*.

Posteriormente, as saídas são comparadas aos valores esperados, uma vez que o *training set* é composto por dados previamente rotulados. Por meio desta comparação, será calculada a função de custo médio ($J(\Omega)$) que será utilizada para ajustar os pesos de todos os neurônios artificiais que compõem a rede. Esse ajuste ocorrerá retroativamente, de modo que, ao se otimizar a última camada, computar-se-á a saída de menor custo para a camada anterior, que também passará pelo processo de otimização e assim sucessivamente até que se chegue na primeira camada da rede (HAYKIN, 2007). Este ajuste é realizado durante a segunda etapa, denominada *backward propagation*, ou simplesmente retropropagação (SILVA; SPATTI, 2010).

A minimização da função de custo médio em algoritmos de ANN durante o *backward propagation* pode ser implementada utilizando o algoritmo de Gradiente Descendente (*Gradient Descent*), que consiste em um processo iterativo que busca a minimização de uma função por meio do ajuste de seus parâmetros. Estes ajustes são necessariamente aplicados na direção de descida mais ingrime, que é sempre oposta ao vetor gradiente e aponta diretamente para um mínimo da função (HAYKIN, 2007). A relação iterativa entre o gradiente de J e atualização da matriz de pesos é expressa por:

$$\Delta\Omega = -\eta \cdot \nabla J, \tag{16}$$

sendo η um valor constante denominado taxa de aprendizagem capaz de indicar o quão rápido o processo de otimização estará se deslocando rumo ao ponto mínimo de J (WIDROW; HOFF, 1960).

Considerando a expressão para cada peso individual que compõe a matriz Ω, o gradiente descendente representado de forma iterativa é representado pela sequinte equação:

$$(\omega_{ij})^{t+1} = (\omega_{ij})^{t} - \eta \cdot \frac{\partial J}{\partial (\omega_{ij})^{t}}. \tag{17}$$

Para que a otimização seja convergente, é necessário que J seja convexa, uma vez que essa condição garante a existência de um único ponto mínimo, denominado mínimo global. Caso essa condição não seja satisfeita, o algoritmo de otimização pode convergir para um ponto de mínimo local, resultando em uma configuração sub-ótima. Além disso, a função de custo pode considerar uma única amostra ou um conjunto reduzido, tendo como finalidade a redução do custo computacional durante o treinamento, porém, a diminuição da complexidade neste processo é alcançada em detrimento de uma maior variabilidade de custo ao longo das épocas de treinamento, uma vez que o ponto de custo mínimo amostral pode divergir do mínimo global (WIDROW; HOFF, 1960). Na literatura, descreve-se J de acordo com a seguinte equação:

$$J(\Omega) = \frac{1}{k} \sum_{j=1}^{k} Cost, \tag{18}$$

em que $Cost$ é o custo individual de cada uma das k amostras que compõem um bloco de treinamento (*training set*) e Ω é a matriz de pesos sinápticos da camada em questão. A função de custo, por sua vez, pode variar de acordo com a ANN implementada.

Dentre as possíveis funções de custo ($Cost$), duas opções são amplamente utilizadas em ANNs: o erro quadrático médio (*Mean Squared Error* - MSE) e a entropia cruzada (*Cross-Entropy* - CE), sendo a primeira aplicável principalmente em problemas de regressão e a segunda para problemas de classificação (ALPAYDIN, 2009). Em particular, as redes neurais descritas no presente trabalho adotam o MSE como função de custo. Desta forma, o custo médio de uma ANN considerando uma camada de saída com m neurônios pode ser descrita da seguinte maneira:

$$J(\Omega) = \frac{1}{2k} \sum_{j=1}^{k} \sum_{i=1}^{m} \left[d_i(j) - y_i(j) \right]^2. \tag{19}$$

em que d_i é a saída desejada e y_i a saída real obtida no i-ésimo neurônio para a j-ésima amostra do *training set*. O fator 2 no denominador é por padrão considerado com a finalidade de simplificação, uma vez que ele será anulado após a diferenciação da função.

Definidas as expressões para as funções de custo ($Cost$) e custo médio (J) ao final da *forward propagation*, o ajuste dos pesos sinápticos pode ser realizado camada a camada durante o *backward propagation*, que se inicia na camada de saída e segue na direção contra propagante até a primeira camada escondida da ANN. O ajuste tem o objetivo de minimizar o erro obtido entre as saídas produzidas (y) e desejadas (d) (SILVA; SPATTI, 2010). Aplicando-se então a definição de gradiente conjuntamente à regra da cadeia, é possível definir ∇J na camada de saída de acordo com a Equação 20.

$$\nabla J = \frac{\partial J}{\partial \Omega^{out}} = \frac{\partial J}{\partial Y_i} \cdot \frac{\partial Y_i}{\partial U_i} \cdot \frac{\partial U_i}{\partial \Omega^{out}}, \tag{20}$$

em que Ω^{out} é a matriz de pesos sinápticos, Y_i o vetor de saída e U_i o vetor potencial de ativação, todos relacionados à camada de saída.

Partindo das Equações 7, 14 e 19 e assumindo que a entrada da camada de saída é igual a saída da última camada escondida ($X_i = Y_{i-1}$), é possível obter as seguintes relações:

$$\frac{\partial J}{\partial Y_i} = -(D_i - Y_i) \tag{21}$$

$$\frac{\partial Y_i}{\partial U_i} = g'(U_i) \tag{22}$$

$$\frac{\partial U_i}{\partial \Omega^{out}} = Y_{(i-1)}, \tag{23}$$

em que D_i é o vetor de saídas esperadas.

Assim, ao substituir as Equações 21, 22 e 22 na expressão 20, obtém-se ∇J da seguinte forma:

$$\nabla J = \frac{\partial J}{\partial \Omega^{out}} = -(D_i - Y_i) \cdot g'(U_j) \cdot Y_{(i-1)} = -\delta_j \cdot Y_{(i-1)}, \tag{24}$$

sendo δ_j, definido como gradiente local, calculado da seguinte forma:

$$\delta_j = (D_i - Y_i) \cdot g'(U_j). \tag{25}$$

Partindo do desenvolvimento matemático descrito, é possível obter o ajuste da matriz de pesos sinápticos por meio do algoritmo de gradiente descendente, apresentado na Equação 16. Assim, podemos definir $\Delta\Omega^{out}$ de acordo com a seguinte equação:

$$\Delta\Omega^{out} = -\eta \frac{\partial J}{\partial \Omega^{out}} = \eta \cdot \delta_j \cdot Y_{(i-1)}. \tag{26}$$

Considerando a Equação 26, é possível otimizar sucessivamente os pesos sinápticos da camada de saída do algoritmo por meio da comparação entre as saídas da ANN e os valores esperados, uma vez que o bloco de treinamento é previamente conhecido. É necessário então otimizar os pesos das camadas anteriores por meio da retropropagação do erro, que é realizada mediante a ponderação do mesmo pelos pesos ajustados. Desta forma, a resposta desejada para neurônios das camadas anteriores é determinada pelos neurônios da camada imediatamente à frente, a qual foi devidamente otimizada no passo anterior (HAYKIN, 2007; SILVA; SPATTI, 2010).

O processo de otimização dos pesos sinápticos das camadas escondidas é semelhante ao descrito para a camada de saída, levando em conta a mesma expressão para ∇J apresentada na Equação 20, porém utilizando matriz Ω e vetores y_i e u_i relacionados à i-ésima camada a ser otimizada. Por intermédio das definições anteriores, têm-se para estes casos as seguintes relações:

$$\frac{\partial J}{\partial Y_i} = \sum_{k=1}^{m} \frac{\partial J}{\partial U_{(i+1)}} \cdot \frac{\partial U_{(i+1)}}{\partial Y_i} = \sum_{k=1}^{m} \frac{\partial J}{\partial U_{(i+1)}} \cdot \frac{\partial \left(\sum_{k=1}^{m} \Omega_{(i+1)} \cdot Y_i \right)}{\partial Y_i} = \sum_{k=1}^{m} \frac{\partial J}{\partial U_{(i+1)}} \cdot \Omega_{(i+1)} \tag{27}$$

$$\frac{\partial Y_i}{\partial U_i} = g'(U_i) \tag{28}$$

$$\frac{\partial u_i}{\partial \Omega} = Y_{(i-1)}, \tag{29}$$

em que m representa a quantidade de neurônios da camada à frente, cujo índice é representado como $(i+1)$. Analogamente ao que foi previamente descrito, é possível definir genericamente o ajuste de pesos sinápticos para a i-ésima camada escondida de acordo com a Equação 30.

$$\Delta\Omega = \eta \cdot \delta_j^{(i)} \cdot Y_{(i-1)}, \tag{30}$$

sendo

$$\delta_j^{(i)} = \left(\sum_{k=1}^{m} \delta_k^{(i+1)} \cdot \Omega_{i+1} \right) \cdot g'(U_j). \tag{31}$$

O procedimento descrito é implementado no *backward* até que se otimize a primeira camada escondida. Cada iteração do processo de *backpropagation*, composto pelas etapas de propagação e contrapropagação, é responsável pelo ajuste gradual dos pesos de modo que, conforme as iterações são realizadas, as matrizes Ω referêntes a cada camada tendem a se aproximar de seus valores ótimos (ALPAYDIN, 2009).

3.2.2 Redes neurais convolucionais

CNNs compõem uma classe de modelos baseada em aprendizado profundo (*deep learning* - DL) cuja principal finalidade é o processamento de dados com padrão em grade (*grid pattern*), tais como imagens e áudio (por meio de histogramas) (YAMASHITA *et al.*, 2018). Este tipo de modelo se inspira na organização do cortex visual de animais (FUKUSHIMA; MIYAKE, 1982) e busca identificar características espaciais de forma automática e adaptativa.

CNNs são formadas por uma construção matemática composta por três tipos de estruturas: camadas convolucionais, *pooling* e rede densamente conectada, sendo a terceira uma rede neural perceptron, já detalhada nas seções anteriores (YAMASHITA *et al.*, 2018). As camadas convolucionais e de *pooling* têm um papel fundamental na extração de características das estruturas de dados (também conhecidas como *feature maps*), ao passo que a rede densamente conectada se encarrega de mapear as características destacadas (ou não) para então gerar uma saída, seja ela de classificação ou regressão (O'SHEA; NASH, 2015).

Tratando-se especificamente de processamento de imagem, uma das principais limitações enfrentadas por ANNs puramente densas (MLPs) é o aumento considerável na complexidade computacional requerida no processamento, uma vez que estas redes possuem

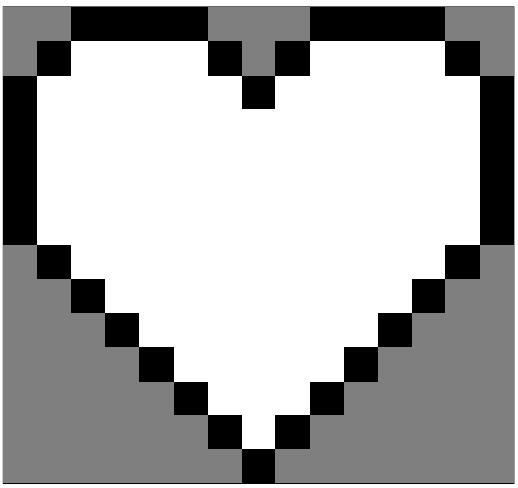
uma grande quantidade de conexões sinápticas. Além disso, a técnica de achatamento (ou *flattening*), que consiste no redimensionamento da imagem para um vetor unidimensional e permite sua inserção na rede, afeta diretamente a capacidade de generalização da ANN, tornando-a sensível à rotações e translações de imagem (O'SHEA; NASH, 2015). CNNs, todavia, mostram-se extremamente capazes de generalizar sua atuação mediante rotações ou translações de padrões, uma vez que, assim como destacado anteriormente, usam estruturas para reconhecer e destacar características antes do processo de *flattening*, utilizando inclusive uma quantidade relativamente pequena de pesos sinápticos por camada convolucional (se comparado às redes densas) (YAMASHITA *et al.*, 2018).

As seções subsequentes apresentarão uma breve descrição relacionada as camadas convolucionais e de *pooling*.

Camadas convolucionais

As camadas convolucionais compõem uma parte fundamental na arquitetura de CNNs: a extração de características, gerando assim os chamados *feature maps*. Esta tarefa é realizada por meio da convolução da entrada com filtros, denominados *kernels*, representado por matrizes quadradas de valores reais. Cada elemento de um *kernel* também é denominado peso, e deve ser um parâmetro otimizável no processo de treinamento (YAMASHITA *et al.*, 2018).

Antes de descrever o processo de convolução, é necessário compreender a forma com a qual imagens são representadas computacionalmente. De modo geral, imagens digitais consistem em uma ou mais matrizes bidimensionais sobrepostas, em que cada elemento se denomina *pixel*, termo proveniente da expressão "*picture element*", ou elemento de figura (FISHER, 1997). Embora existam diferentes padrões para representação de cores, um dos mais utilizados é o RGB, cuja sigla é a descrição das cores vermelho, verde e azul (***R**ed, **G**reen and **B**lue*). Normalmente, os valores em cada escala de cor (1 escala para figuras monocromáticas ou 3 para figuras coloridas em RGB) são representados por um byte, ou 8 bits, podendo então assumir valores entre 0 e 255. Nesta escala representa-se a intensidade de luz de um determinado pixel, sendo 0 a ausência de luz e 255 a intensidade máxima (STOKES, 1996). A Figura 9 apresenta uma imagem em escala monocromática de resolução 14x15 e sua respectiva representação na forma matricial.



127	127	0	0	0	0	127	127	127	0	0	0	0	127	127
127	0	255	255	255	255	0	127	0	255	255	255	255	0	127
0	255	255	255	255	255	255	0	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	255	255	255	255	255	0
127	0	255	255	255	255	255	255	255	255	255	255	255	0	127
127	127	0	255	255	255	255	255	255	255	255	255	0	127	127
127	127	127	0	255	255	255	255	255	255	255	0	127	127	127
127	127	127	127	0	255	255	255	255	255	0	127	127	127	127
127	127	127	127	127	0	255	255	255	0	127	127	127	127	127
127	127	127	127	127	127	0	255	0	127	127	127	127	127	127
127	127	127	127	127	127	127	0	127	127	127	127	127	127	127

Figura 9 – Representação computacional de uma imagem monocromática de dimensão 14x15. A matriz na direita apresenta valores entre 0 e 255, que corresponde à intensidade de um pixel de 8 bits.

Em CNNs, definem-se tensores como sendo matrizes numéricas (normalmente imagens) que são inseridas na entrada de uma camada convolucional. Durante a convolução, um ou mais *kernels* executam uma varredura sobre o tensor com um passo pré-determinado. Em cada um desses passos, os pesos que compões cada *kernel* são multiplicados pelos elementos do tensor e posteriormente somados, gerando assim combinações lineares de uma determinada região da matriz de entrada. Trata-se portanto de uma convolução em duas dimensões, sendo cada *kernel* a representação espacial da função de resposta ao impulso de um filtro, responsável por destacar alguma determinada característica na imagem. Essas combinações, que são análogas ao potencial de ativação em camadas densas de redes neurais perceptron, são então aplicadas a uma função de ativação que deve limitar a faixa de valores da saída (O'SHEA; NASH, 2015; KOUSHIK, 2016), compondo assim um determinado *feature map*. É necessário lembrar que pixels não podem ser representados por números negativos, portanto, uma das funções de ativação mais utilizadas para processamento de imagens é a ReLU (descrita na Seção 3.1.3), que possui comportamento linear para valores positivos e saída nula para negativos (XU *et al.*, 2015). O processo de convolução aqui descrito pode ser analisado com maiores detalhes na Figura 10.

Um dos principais problemas relacionados a operação de convolução, descrita acima, é o fato de que ela não possibilita que pesos no centro de cada *kernel* se sobreponham

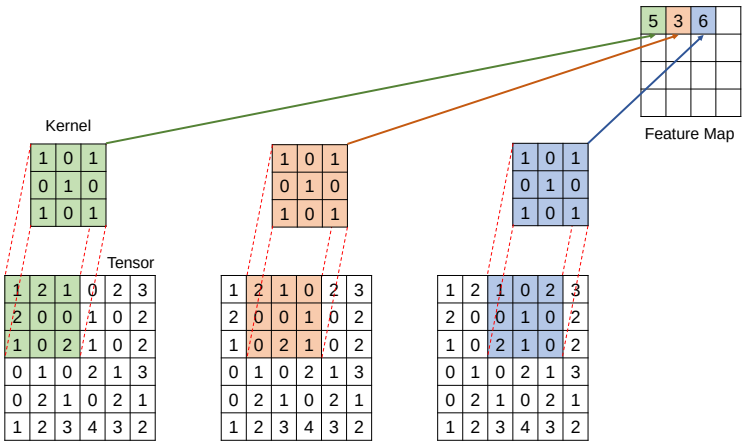


Figura 10 – Exemplo de convolução de um *kernel* de dimensão 3x3 com um *tensor* de 6x6 e varredura de passo 1. O *feature map* resultante possui dimensão 4x4.

aos pixels das extremidades. Sendo assim, caso hajam características relevantes próximas às bordas de um tensor que devam ser destacadas em um determinado *feature map*, elas podem acabar sendo pouco consideradas (WU, 2017). Uma das técnicas mais utilizadas para resolver este problema é denominada *padding*, que consiste na inserção de linhas e colunas ao redor da imagem, que podem garantir que todos os pixels sejam considerados igualmente durante as operações. Normalmente as bordas adicionadas no *padding* possuem conteúdo nulo (denominando-se assim *zero-padding*), e podem conter mais de uma camada, dependendo das dimensões do *kernel* (WU, 2017; YAMASHITA *et al.*, 2018). Como exemplo, a Figura 11 apresenta uma imagem antes e após a aplicação de um *zero-padding* de duas camadas.

O processo de convolução tem como principal objetivo destacar determinados tipos de características que sejam importantes para realizar a tarefa desejada, seja ela uma regressão ou classificação. O resultado obtido ao realizar tal procedimento está diretamente relacionado com os pesos de que compõem o *kernel* em questão. Podem ser realisadas tarefas como detecção de borda e/ou determinados formatos geométricos, destaque de relevo (*embossing*), suavização de detalhes, etc. (JUNG; SHIN; KWON, 2018) As matrizes 3×3 a seguir, representadas pelas Equações 32 e 33, representam exemplos de *kernels* de destaque de relevo e detecção de borda, respectivamente.

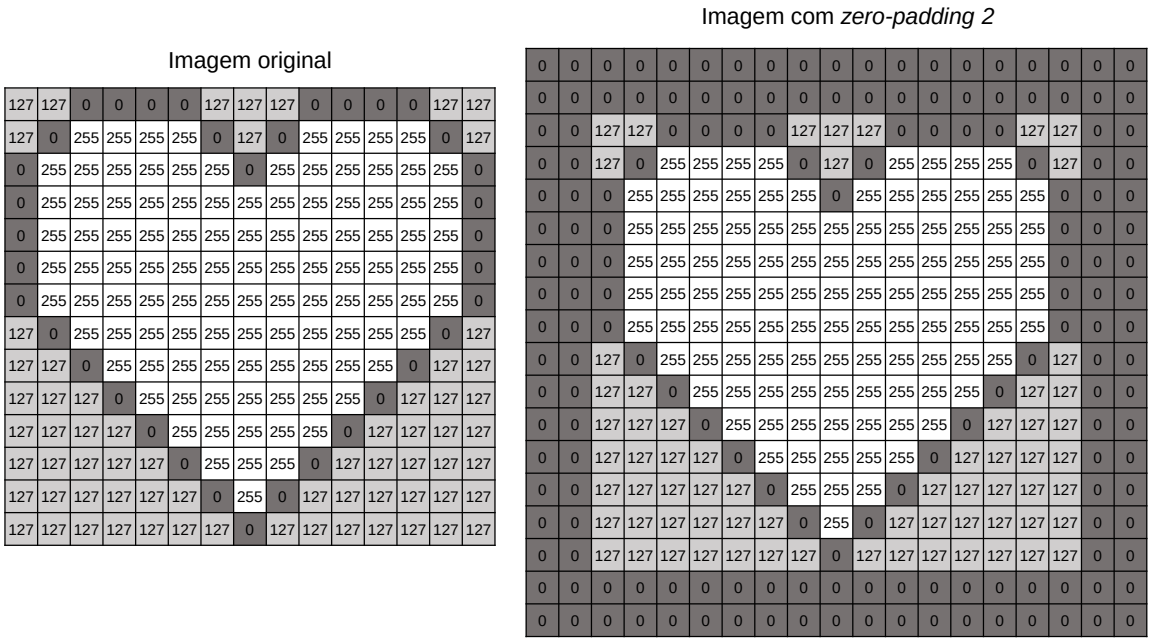
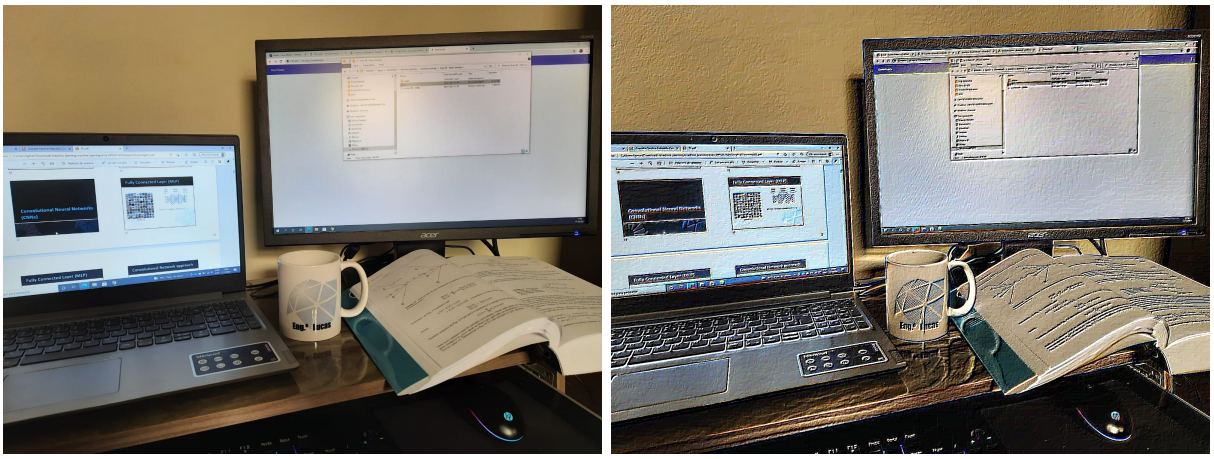


Figura 11 – Exemplo de imagem com *zero-padding* de duas camadas.

$$\begin{bmatrix} -9 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 9 \end{bmatrix} \tag{32}$$

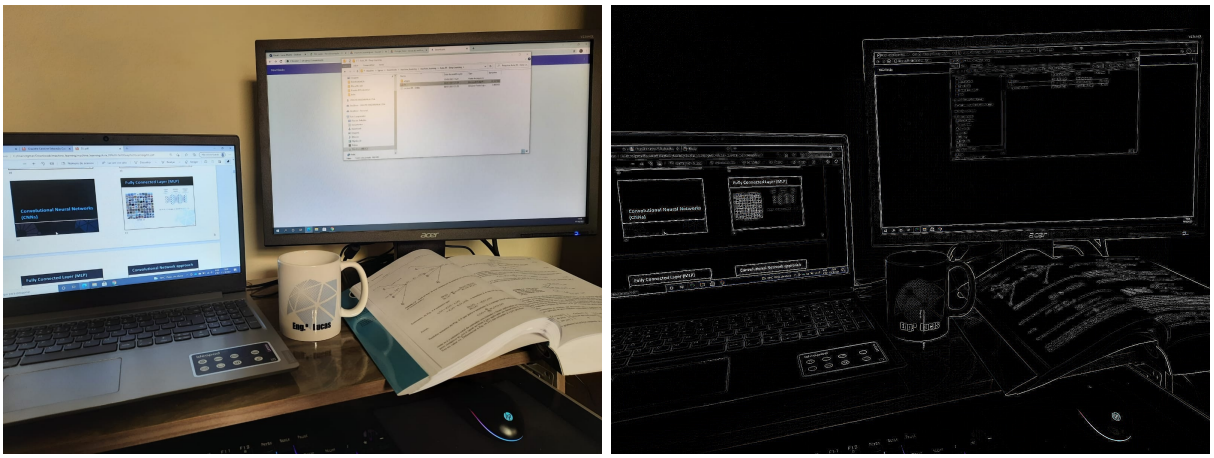
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{33}$$

A Figura 12 apresenta uma imagem genérica capturada pelo autor e o resultado obtido ao convoluí-la com o *kernel* apresentado na Equação 32, cuja principal finalidade é destacar relevos na imagem. Já a Figura 13 mostra novamente a mesma imagem originalmente capturada pelo autor e o resultado obtido por meio da convolução da figura com o *kernel* representado pela Equação 33, cuja principal tarefa é a detecção de bordas.



(a) (b)

Figura 12 – (a) Imagem demonstrativa capturada pelo autor. (b) Imagem obtida após a convolução da imagem original com o *kernel* representado pela Equação 32, responsável por destacar regiões de profundidade (embossing).



(a) (b)

Figura 13 – (a) Imagem demonstrativa capturada pelo autor. (b) Imagem obtida após a convolução da imagem original com o *kernel* representado pela Equação 33, responsável por detectar bordas.

É válido lembrar que, embora os *kernels* possam ser utilizados para destacar determinadas características objetivamente perceptíveis à observação, os pesos que compões os *kernels* em camadas convolucionais são determinados por meio de um processo de otimização puramente estatístico, cujo objetivo fundamental se resume à redução de uma determinada função de custo atribuída a um conjunto de dados de treinamento. Sendo assim, as características destacadas em camadas convolucionais podem de alguma forma ser contraintuitivas caso observadas de forma isolada (características subjetivas), porém destacam elementos que foram estatisticamente relevantes para realizar a tarefa desejada (O'SHEA; NASH, 2015).

Camadas de *pooling*

Além das camadas convolucionais, CNNs também são compostas pelas chamadas camadas de *pooling*, cuja principal tarefa é a redução de dimensão de *feature maps*, gerando consequentemente uma redução na quantidade de parâmetros de ativação dentro da rede (O'SHEA; NASH, 2015).

O processamento realizado por essas camadas se assemelha em parte ao das camadas convolucionais, pois nela são considerados filtros que varrem a imagem com um passo pré-determinado (*stride*). Todavia, a operação realizada é sumariamente diferente, pois ao invés de se convoluir a entrada da camada a um filtro, o *pooling* basicamente realiza uma operação com os próprios pixels da imagem, como por exemplo o cálculo da média (*average pooling*) ou a seleção do elemento de maior valor (*max pooling*) (WU, 2017).

Desta forma, a saída de uma camada de *pooling* é uma representação da entrada, porém com dimensão reduzida. A configuração mais utilizada na prática é o *max pooling* com filtro de dimensão 2×2 e *stride* (passo de varredura) 2, porém o *average pooling* também se mostra bastante relevante para algumas arquiteturas específicas de CNNs (YAMASHITA *et al.*, 2018).

Estrutura geral e diferentes arquiteturas de CNNs

Assim como já descrito, CNNs são compostas por três tipos de estruturas: as camadas convolucionais, camadas de *pooling* e rede densamente conectada (rede neural

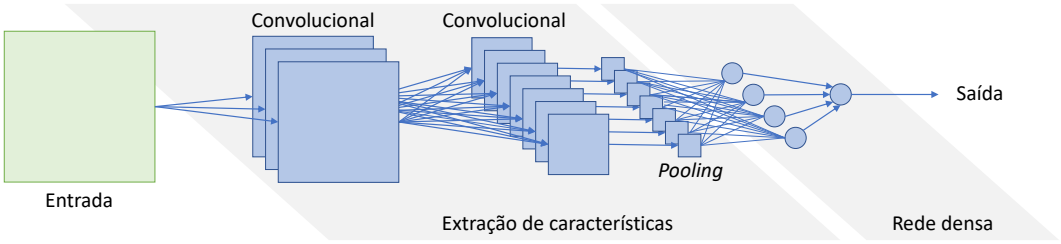


Figura 14 – Estrutura geral de uma rede neural convolucional

artificial *multilayer* perceptron). Quando utilizada para processamento de imagem, este tipo de rede normalmente recebe três matrizes (um tensor) de entrada, sendo cada uma a representação de um canal de cor (R, G e B) (WU, 2017). A primeira etapa da rede consiste na extração de características, que se dá por meio das camadas convolucionais. Com a finalidade de diminuir a quantidade de parâmetros otimizáveis (e consequentemente o custo computacional), os processos de redução de dimensionalidade dos *feature maps* são realizados por meio da utilização de camadas de *pooling*, que normalmente fazem uso da técnica de *max pooling*. Após obter uma grande quantidade de características mapeadas, os *feature maps* são rearranjados na forma de vetores unidimensionais no processo conhecido como *flattening*, e então inseridos em uma rede perceptron, que deverá processá-los para então gerar uma ou mais saídas desejadas, de acordo com o tipo de tarefa a ser realizada (O'SHEA; NASH, 2015). Esta arquitetura de rede é mostrada na Figura 14.

De forma análoga às ANN perceptrons, o treinamento de CNNs é performado pela otimização dos pesos sinápticos das diversas camadas que compões a rede, sejam elas densas ou convolucionais. Sendo assim, podem-se utilizar os mesmos procedimentos de otimização descritos na Subseção 3.2.1, adotando-se inclusive o gradiente descendente para otimizar a função de custo, que novamente deve ser escolhida de acordo com a tarefa a ser realizada. Durante o treinamento da CNNs, o algoritmo de *back propagation* é novamente aplicado propagando-se os dados do conjunto de treinamento até a saída, computando a função de custo e retropropagando o erro para que se possa otimizar os pesos sinápticos que compõem a rede. Embora camadas convolucionais sejam estruturalmente diferentes das

camadas densas, elas são ainda assim compostas fundamentalmente por pesos sinápticos, que podem ser otimizados com a finalidade de se reduzir a função de custo.

O estudo e desenvolvimento de diferentes arquiteturas de CNNs tem sido responsável por gerar modelos cada vez mais sofisticados e eficientes, que baseiam-se em métricas comparativas para validar seu funcionamento frente a outros modelos. Uma das métricas mais utilizadas para análise de desempenho de CNNs é a classificação de imagens partindo de *datasets* pré definidos, que contam com centenas de milhares de imagens, como é o caso do ImageNet. Em particular, o dataset *ImageNet* é construído baseado em uma estrutura hierárquica que contém na ordem de 50 milhões de imagens rotuladas e divididas em categorias relacionadas. Partindo do *dataset*, duas métricas de avaliação de desempenho são comumente utilizadas: a top-5, que considera acerto caso o rótulo correto esteja entre as 5 opções mais prováveis apontadas pela rede, e a top-1, que considera somente classificações categoricamente corretas (DENG *et al.*, 2009).

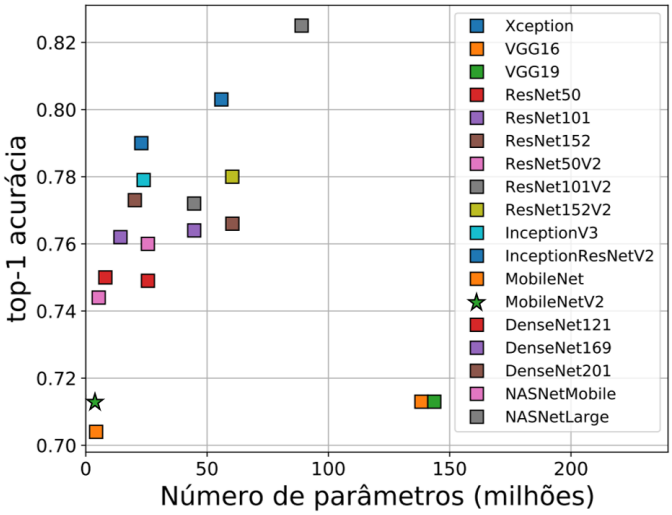


Figura 15 – Desemprenho de alguns modelos de CNNs de acordo com o modelo de desempenho top-1 baseado no *dataset* de validação ImageNet

A Figura 15 apresenta o desempenho de algumas arquiteturas de CNNs no teste de classificação top-1 baseados no *dataset* ImageNet. Na imagem, o eixo horizontal representa a quantidade de parâmetros otimizáveis de cada arquitetura, que está diretamente relacionada ao custo computacional das diferentes CNNs para classificar as imagens. Dentre as opções apresentadas, a MobileNetV2 se apresenta como sendo um bom modelo a ser adotado devido ao *tradeoff* entre complexidade e desempenho.

3.3 Analise de componentes principais

Assim como discutido, as camadas de *pooling* e de convolução auxiliam na extração de padrões em conjuntos de dados de alta dimensionalidade, tais como imagens. Entretanto, em certas aplicações é ainda necessário utilizar mecanismos para redução de dimensionalidade. Neste contexto, a análise de componentes principais (PCA - principal component analysis) tem sido amplamente utilizada para esta finalidade, sendo tratada na literatura como uma ferramenta pertencente ao grupo de algoritmos de aprendizagem auto-organizada. Trata-se de um mecanismo não supervisionado que busca identificar padrões significativos em um determinado bloco de dados (HAYKIN, 2007).

Considerando um *dataset* composto por dados de diferentes características, o PCA mostra-se eficaz na tarefa de identificar parâmetros responsáveis pela maior variância do sistema, ou seja, as características que são particularmente mais impactantes na sensibilidade dos dados de saída. Essas características são descritas como combinações lineares dos parâmetros iniciais do bloco de dados (HAYKIN, 2007).

A redução de dimensionalidade por meio do PCA considera, dentro do contexto da álgebra linear, os conceitos de autovalores e autovetores. Tem-se que a direção que concentra a maior variância de um sistema de dimensão n coincide com a de um dos n autovetores do sistema (ALPAYDIN, 2009), assim como exemplificado pelo gráfico da Figura 16.

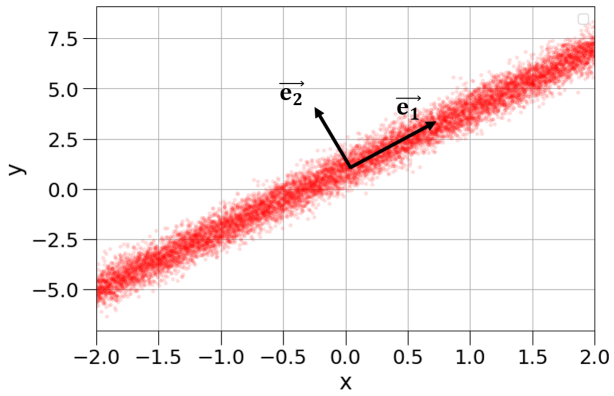


Figura 16 – Representação no plano cartesiano de um conjunto de dados com 2 parâmetros e seus respectivos autovetores $\vec{e}_1$ e $\vec{e}_2$.

Partindo da representação gráfica da Figura 16, nota-se que a direção responsável pela maior variância dos dados coincide com a do autovetor $\vec{e}_1$, e que portanto o sistema,

que inicialmente apresenta 2 características (bidimensional), pode ser representado por uma curva unidimensional, ou seja, uma reta.

O percentual da variância total nas direções de $\vec{e}_1$ e $\vec{e}_2$ pode ser calculada por meio da determinação dos respectivos autovalores, conforme a Equação 34.

$$\%PC_i = \frac{\lambda_i}{\sum \lambda_j} \times 100 \tag{34}$$

sendo PC_i a componente principal (PC - *principal component*) na direção do i-ésimo autovetor e λ_i o seu respectivo autovalor (ALPAYDIN, 2009).

Embora não seja obrigatória, a aplicação do PCA para redução de dimensionalidade normalmente é subsequente à normalização dos dados. Para se obter os autovalores e autovetores de um determinado bloco de dados, é inicialmente necessário calcular a matriz de covariância, cujos elementos representam a relação de variabilidade entre os diferentes parâmetros do sistema (HAYKIN, 2007). Como exemplo, podemos considerar um *dataset* em que cada amostra possui informação de três parâmetros, representados por a, b e c. Neste caso, a matriz de covariância pode ser representada pela Equação 35:

$$\Sigma = \begin{bmatrix} var(a) & cov(a,b) & cov(a,c) \\ cov(b,a) & var(b) & cov(b,c) \\ cov(c,a) & cov(c,b) & var(c) \end{bmatrix} \tag{35}$$

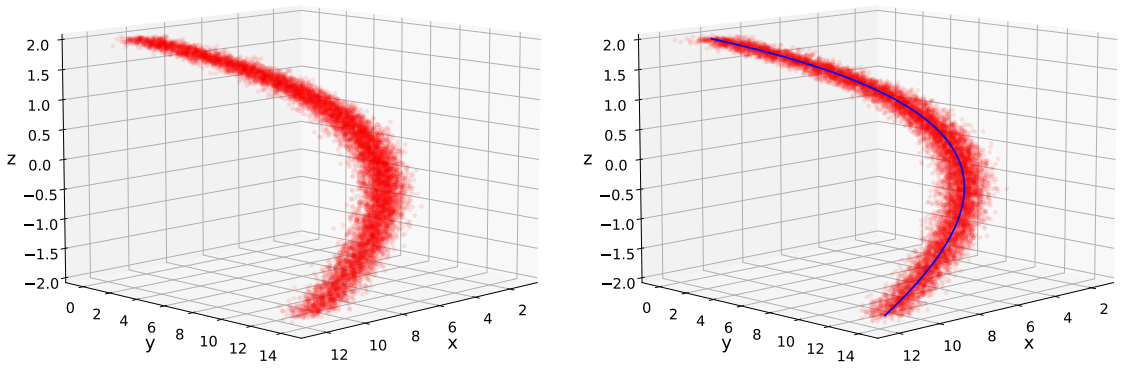
em que $var(a)$ é a variância do a-ésimo parâmetro e $cov(a,b)$ a covariância amostral entre os parâmetros a e b, que para um bloco de n amostras é representada pela Equação 36.

$$cov(x,y) = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{(n-1)} \tag{36}$$

Considerando uma matriz covariância Σ, sabe-se que a expressão que a relaciona aos autovalores λ e autovetores $\vec{e}$ é descrita pela Equação 36.

$$A \cdot \vec{e} = \lambda \cdot \vec{e} \tag{37}$$

Desta forma, podem-se obter λ e $\vec{e}$ por meio das respectivas equações:



(a) (b)

Figura 17 – (a) Representação de conjunto de dados com descritos pelos parâmetros x, z e z . (b) Representação do bloco de dados conjuntamente à curva de tendência das amostras.

$$det(\Sigma - \lambda I) = 0 \tag{38}$$

e

$$(\Sigma - \lambda I) \cdot \vec{e} = \vec{0} \tag{39}$$

sendo I a matriz identidade.

Após a obtenção dos autovalores e autovetores de um conjunto de dados, o PCA possibilita a análise das componentes que possuem maior representatividade na variância total dos parâmetros. A aplicação de critérios para selecionar o número de componentes a serem consideradas seguida de uma transformação de base resulta na redução de dimensionalidade do problema (ALPAYDIN, 2009).

Como exemplo, podemos considerar um conjunto de dados de 10.000 amostras com parâmetros x, y e z que seguem a tendência de uma curva parabólica, assim como apresentado na Figura 17. Por se tratar de uma cônica, a representação gráfica da parábola está necessariamente contida em um plano (bidimensional), que por sua vez é representado por uma base ortonormal cujos versores apontam na mesma direção dos dois autovetores mais representativos.

Para este exemplo, aplicaremos o PCA após a normalização dos dados, assim como mostrado na Figura 18. Os dados originais são centralizados e padronizados de acordo com as seguintes equações:

$$x' = \frac{x - \mu_x}{\sigma_x} \tag{40}$$

$$y' = \frac{y - \mu_y}{\sigma_y} \tag{41}$$

$$z' = \frac{z - \mu_z}{\sigma_z} \tag{42}$$

sendo μ e σ os valores médios e desvios de cada uma das respectivas componentes. A padronização é seguida da normalização, que se encarrega que limitar os valores em cada uma das componentes no intervalo de [0, 1].

Após calcular e analisar os autovalores e autovetores, apresentados na Tabelas 1, é possível notar que a variância nas direções de $\vec{e}_1$ e $\vec{e}_2$ somadas correspondem a 99,63 % da variância total dos dados. Sendo assim, o PCA evidencia a possibilidade de redução da dimensionalidade do conjunto para uma base bidimensional.

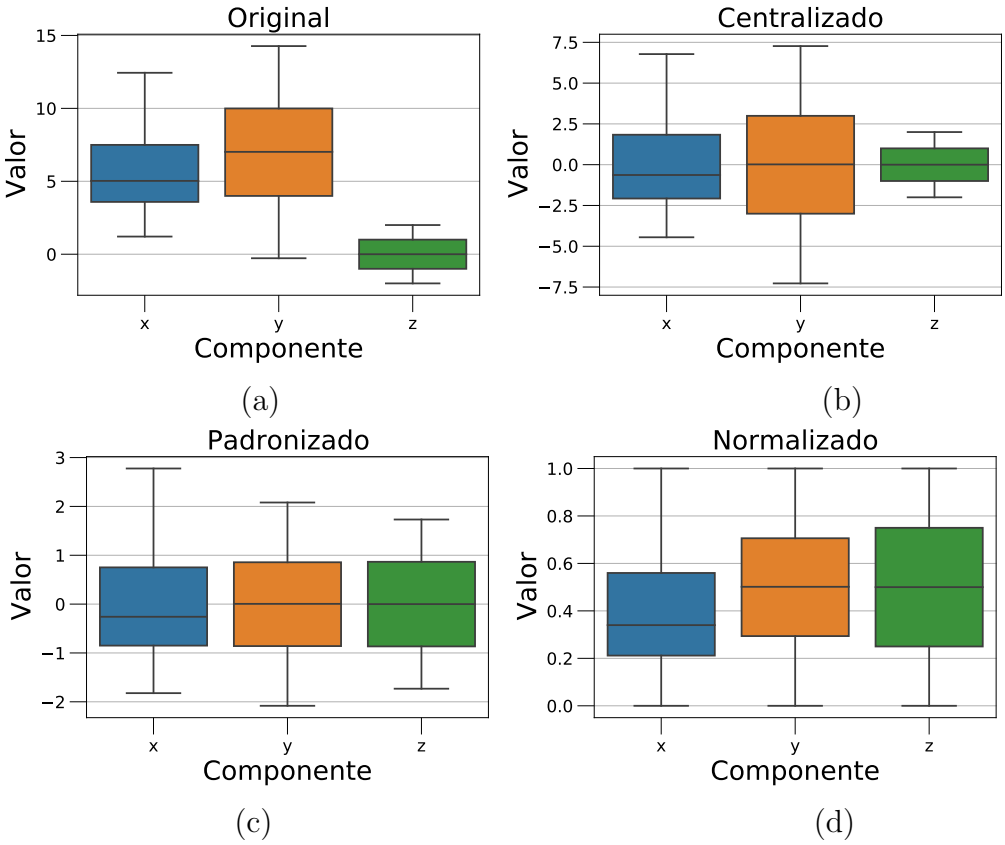


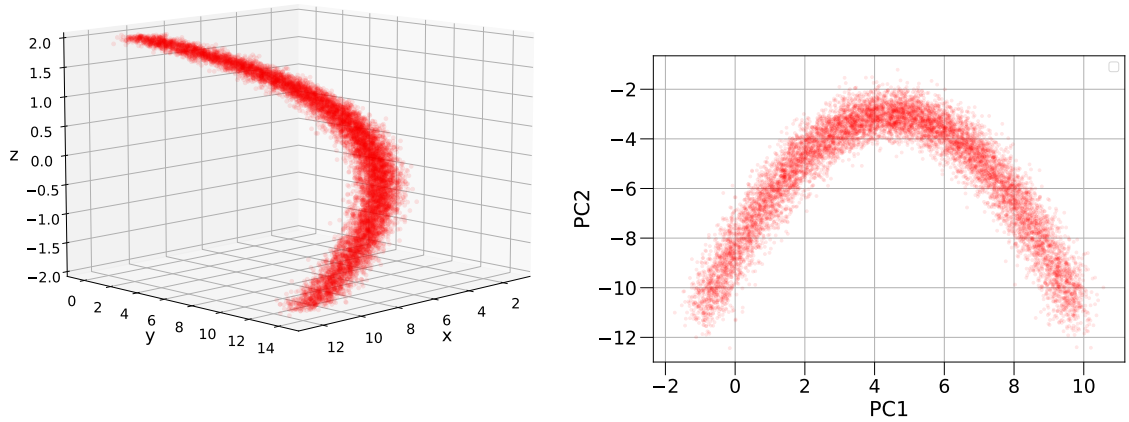
Figura 18 – Representação gráfica do intervalo e variância das amostras nas direções x, y e z dos bloco de dados (a) original, (b) após centralização, (c) padronização e (d) e normalização.

Tabela 1 – Autovetores, autovalores e participação percentual das diferentes componentes na variância total dos dados de exemplo

	Autovetores	Autovalores	% na variância
$\vec{e}_1$	(-1.00000, -0.00037, 0.00272)	0.047281	25.10
$\vec{e}_2$	(-0.00233, 0.63899, -0.76921)	0.140407	74.53
$\vec{e}_3$	(0.00145, 0.76921, 0.63898)	0.000689	0.37

Tabela 2 – Versores da nova base (bidimensional)

Versores das componentes principais	
$\overrightarrow{PC}_1$	(-0.002332, 0.638990, -0.769211)
$\overrightarrow{PC}_2$	(-0.999996, -0.000371, 0.002723)



(a) (b)

Figura 19 – Representação do conjunto de dados de exemplo nas bases (a) original (tridimensional) (b) e de dimensão reduzida (bidimensional).

Neste exemplo, os versores $\overrightarrow{PC}_1$ e $\overrightarrow{PC}_2$ que compõem a nova base são então calculados e apresentados na Tabela 2. A Figura 19 apresenta uma comparação entre as representações dos dados de exemplo na base inicial (tridimensional) e na base nova (bidimensional). Neste caso a variância total do conjunto é similar em ambos os casos, uma vez que a componente desconsiderada (na direção de $\vec{e}_3$) representa apenas 0,37 % da variância total.

O presente exemplo demonstra que o PCA não somente possibilita a redução de dimensionalidade, mas também fornece ferramentas para análise de dispersão de dados.

Particularmente para este trabalho, estas ferramentas foram utilizadas para analisar a dispersão das estimativas de BER performadas por CNNs em direções associadas ao erro de predição.

4 Arranjo de simulação

4.1 Especificações técnicas do sistema de comunicação óptica considerado

Para este trabalho, consideram-se inicialmente sistemas de comunicações ópticas coerentes digitais semelhantes ao mostrado no esquemático da Figura 1, com enlaces de diferentes comprimentos. Sendo assim, os sistemas representam redes de distribuição LR-PON de dupla polarização com mecanismos de compensação de ruído de fase e dispersão cromática por meio de DSP. As principais especificações técnicas dos sistemas considerados encontram-se descritas abaixo:

- **Comprimento:** Enlaces *single span* de 80, 90, 100, 110, 120, 130, 140 e 150 km;
- **Taxa de transmissão:** 50 Gbps por polarização, totalizando uma taxa de 100 Gbps no canal;
- **Largura de linha dos lasers:** 100 kHz;
- **Formato de modulação:** 16 QAM mapeado utilizando codificação de Gray;
- **Potência de transmissão (*Launch optical power*, LOP):** De 0 a 13 dBm (passo de 1 dB) para cada um dos diferentes comprimentos de enlace;

Os sistemas foram implementados computacionalmente no software *VPI Transmission Maker* e simulados considerando um total de 16284 símbolos (65136 bits) para cada valor de potência especificado. Os símbolos recebidos foram classificados por meio de detecção por máxima verossimilhança e os valores de BER estimados por meio de contagem de erros.

4.2 Especificações da arquitetura de rede convolucional, parâmetros de rotulação e treinamento

Com a finalidade de realizar a predição da BER em sistemas de comunicações ópticas com as especificações previamente apresentadas, propôs-se utilizar redes neurais convolucionais baseadas na arquitetura MobileNetV2 para regressão, considerando como entrada e saída histogramas de constelações de sinais detectadas no receptor e a estimativa da BER, respectivamente.

A arquitetura de CNN conhecida como MobileNetV2 trata-se de um modelo composto por um total de 3.538.984 parâmetros otimizáveis. Esta arquitetura é majorita-

riamente composta por camadas convolucionais, contando apenas com uma camada de *average pooling* seguida de uma camada densa na saída. Ao longo de sua estrutura, são dispostas consecutivamente camadas convolucionais com *kernels* de dimensão 1×1 e 3×3, que somadas são responsáveis por 75,67% dos parâmetros otimizáveis da rede (HOWARD *et al.*, 2017). Esta arquitetura de rede foi inicialmente proposta em 2017, apresentando desempenho de 0,713 e 0,901 nos testes top-1 e top-5 de classificação utilizando o *dataset ImageNet*, respectivamente. A performance obtida é similar à das redes VGG-16 e VGG-19, que possuem mais de 138 milhões de parâmetros otimizáveis (SIMONYAN; ZISSERMAN, 2014).

Sendo assim, os principais motivos da escolha da MobileNetV2 foram o seu custo computacional reduzido, se comparado às principais arquiteturas de CNNs, e seu desempenho satisfatório em métricas formais comparativas para este tipo de algoritmo. Embora originalmente esta rede esteja associada a uma camada de entrada de dimensão 224×224×3, realizou-se uma redução no tamanho do tensor inicial para 128×128×3, uma vez que os histogramas de constelações processados teriam número de *bins* consideravelmente menores que as dimensões de entrada originais da rede (224). Esta redução por si só gerou uma diminuição na quantidade total de parâmetros da rede, que passou de 3.538.984 para um total de 2.259.265 pesos.

Os modelos de CNNs foram implementados em linguagem *python* por meio das bibliotecas *Keras* e *Tensorflow*, amplamente utilizadas para a implementação de modelos dos mais diferentes tipos de redes neurais artificiais. A estrutura geral das redes MobileNetV2 original e modificada (de acordo com as características apresentadas) podem ser analisadas nas tabelas em anexo, que apresentam cada uma das camadas que compõem a rede, bem como a dimensão de saída e quantidade de parâmetros que por camada. A análise de complexidade dos modelos será abordada após a discussão de resultados, na seção 5.5. Esta análise apresentará o custo computacional (quantidade operações de ponto flutuante) necessário para realizar uma estimativa de BER por meio do processamento de um histograma de constelação.

4.3 Especificações do bloco de dados e métricas de treinamento

Para realizar o treinamento e validação das CNNs, realizou-se um tratamento de dados para organizar e rotular os resultados simulados. Para isso, foram gerados histogramas das constelações obtidas no receptor dos sistemas ópticos com quantidade de símbolos e bins variáveis.

É necessário lembrar que idealmente a quantidade de símbolos necessária para estimar a BER deve ser a menor possível, pois a exigência de uma sequência piloto cada vez menor resulta em uma menor latência para a predição. Sendo assim, para o treinamento foram gerados diferentes *datasets* com histogramas compostos por quantidades de símbolos que variam de 1.000 a 10.000, com passo 1.000. É importante destacar que estes diferentes blocos de dados foram processados separadamente, pois um dos parâmetros importantes para a avaliação do sistema de predição é o desempenho em função da quantidade de símbolos contidos na sequência piloto.

Outro parâmetro importante a ser considerado neste tipo de sistema é a quantidade de *bins* dos histogramas, uma vez que as CNNs estão sendo utilizadas para o processamento de imagem. Geraram-se, portanto, *datasets* de imagens quadradas considerando diferentes *bins*, com valores que variam de 32 a 64 e passo 4.

Considerando as diferentes quantidades de símbolos (10 valores) e *bins* (9 tamanhos) dos histogramas, geraram-se um total de 90 *datasets*, cada um composto por 6.000 histogramas. Para cada bloco de dados, foram gerados histogramas de constelações que levaram em conta enlaces com diferentes comprimentos e potência de transmissão (descritos na seção anterior), buscando manter quantidades balanceadas de amostras para os diversos valores de BER. Sendo assim, cada *dataset* possui quantidades semelhantes de constelações de sistemas limitados por ruído gaussiano e por não linearidades, pois a variedade e homogeneidade são fatores necessários para se garantir uma boa capacidade de generalização da rede.

Um fato importante a ser destacado com relação à rotulagem dos dados é que ela foi performada por meio da contagem de erros de bit do bloco total de 16.284 símbolos simulados (para cada sistema simulado), pois ao considerar amostras pequenas (1.000, 2.000, 3.000 símbolos, etc), a probabilidade de erro de bit pode ser subestimada. Como exemplo, caso se considerem apenas 1000 símbolos, a menor quantidade de erro de símbolo

possível de ser estimada utilizando contagem de erros é de 1 em 1000. Sendo assim, cada histograma foi rotulado com a quantidade de erros computada no bloco total de símbolos simulados, e não no valor contado considerando somente a amostra reduzida que gerou o histograma.

Os histogramas foram gerados em linguagem *python* por meio da biblioteca *matplotlib*, utilizando o *colormap twilight*. Para garantir a utilização de um mesmo modelo de CNN para cada um dos 90 diferentes *datasets*, fixou-se a dimensão de entrada da imagem em 128×128×3. Sendo assim, todos os histogramas foram redimensionados para esta resolução sem a aplicação de interpolação de *pixels* ou qualquer outro tipo de técnica de suavização. Desta forma, mantiveram-se as características originais das figuras para que se pudesse validar o desempenho do algoritmo em função somente do número de *bins* inicial dos histogramas.

A Figura 20 apresenta uma mesma constelação representada por diferentes valores de *bins* adotados no tratamento de dados previamente descrito. Já a Figura 21, apresenta diferentes histogramas de constelações considerando diferentes quantidades de símbolos. Para todos os casos considera-se um mesmo sistema operando com um único valor de potência.

Após obter os blocos de dados, definiram-se as métricas de treinamento para as redes convolucionais aplicadas a cada uma das diferentes combinações de *bins* e quantidade de símbolos. Considerando uma quantidade total de 6000 imagens por *dataset*, os blocos de dados de treinamento e validação cruzada foram divididos na proporção 0,7/0,3, ou seja, 70% dos histogramas foram alocados para treino e os outros 30% para validação. O grupo de validação cruzada tem papel fundamental para evitar a ocorrência de *overfitting*, e não é considerado pelo algoritmo de otimização, que neste caso é o gradiente descendente de lote. Após realizar o treinamento e selecionar modelos com base no desempenho obtido pela rede no bloco de validação, as CNNs foram submetidas a uma etapa de teste que considerou outros 1000 histogramas para cada um dos 14 valores de potência de transmissão, considerando um comprimento de enlace óptico passivo pre-definido (de 150 km). Sendo assim, o bloco de dados de teste compôs-se de 14.000 amostras.

A inicialização dos pesos na rede foi realizada de forma aleatória seguindo uma distribuição gaussiana de média 0 e desvio padrão unitário, e as funções de ativação adotadas foram a ReLU para camadas convolucionais e linear para o neurônio de saída, uma vez que se trata de uma rede de regressão.

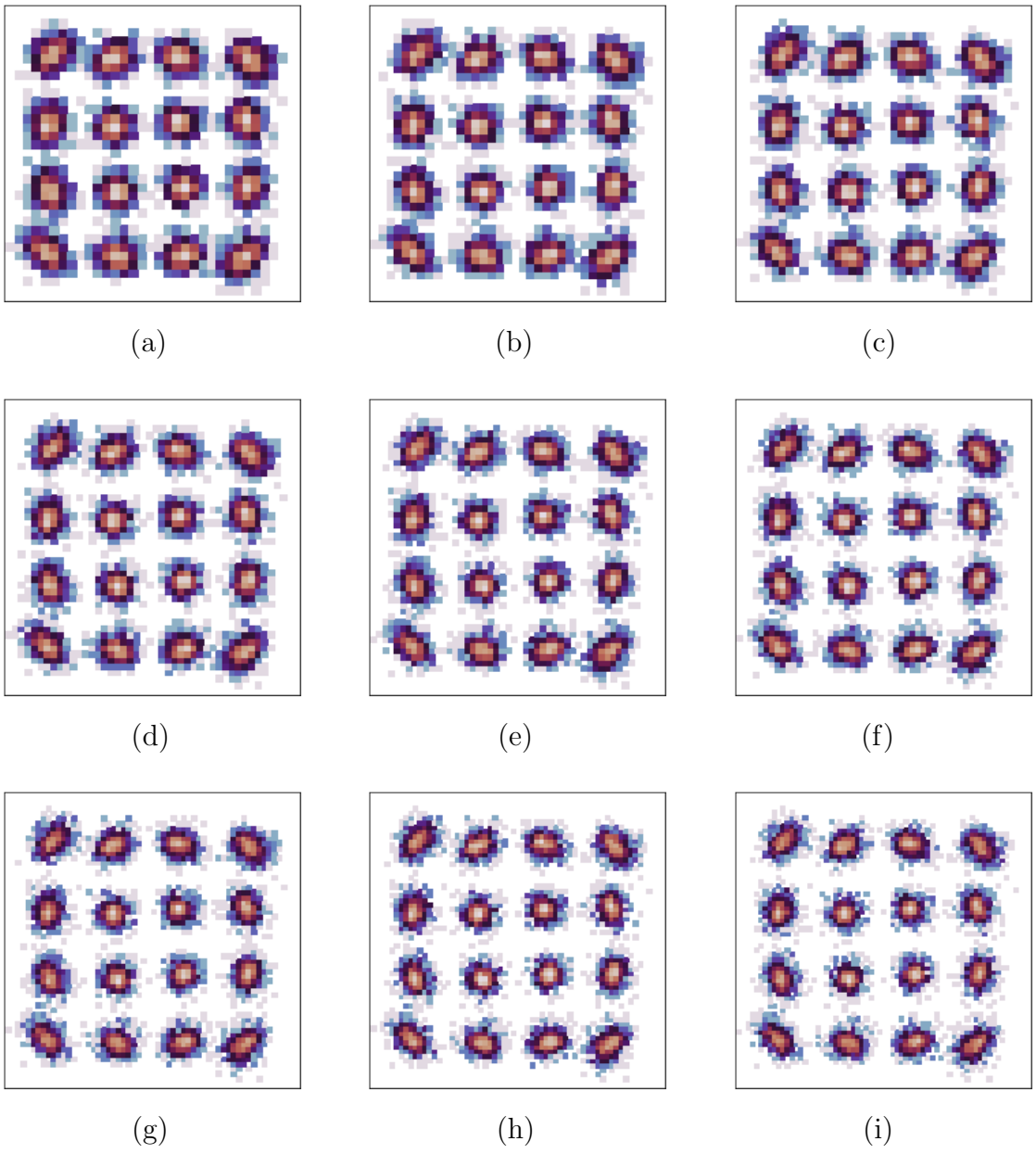


Figura 20 – Histogramas da constelação recebida (16-QAM) com 16284 símbolos em um sistema óptico passivo *single channel* de 80 km de comprimento para potência transmitida de 9 dBm com dimensão lateral de (a) 32, (b) 36, (c) 40, (d) 44, (e) 48, (f) 52, (g) 56, (h) 60 e (i) 64 *pixels*. Valores de amplitude em unidades arbitrárias.

A etapa de treinamento foi realizada considerando o algoritmos de gradiente descendente (Seção 3.2.1) para subgrupos de treino (*batch size*) de 50 imagens. Foram utilizadas 300 épocas de treinamento para cada uma das 90 diferentes CNNs. Os resultados obtidos poderão ser analisados nas seções subsequentes.

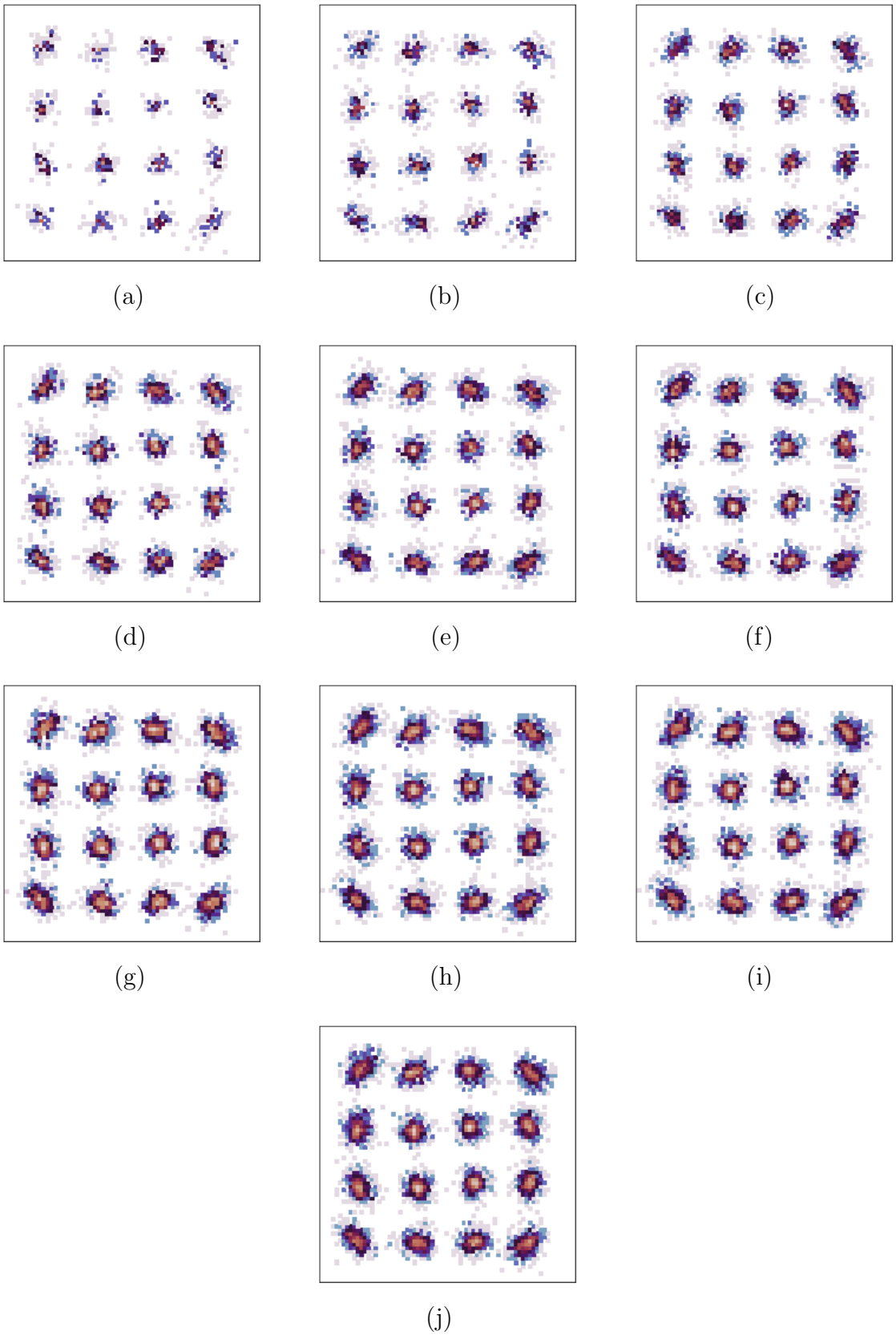


Figura 21 – Histogramas com resolução 64×64 da constelação recebida (16-QAM) em um sistema óptico passivo *single channel* de 80 km de comprimento para potência transmitida de 9 dBm considerando (a) 1000, (b) 2000, (c) 3000, (d) 4000, (e) 5000, (f) 6000, (g) 7000, (h) 8000, (i) 9000 e (j) 10000 símbolos. Valores de amplitude em unidades arbitrárias.

5 Resultados

Nessa seção serão apresentados os principais resultados obtidos após as simulações e treinamento das diferentes CNNs.

5.1 Análise da probabilidade de erro de bit estimada por EVM para diferentes cenários

Para validar a necessidade da utilização de novos métodos para a predição da BER, inicialmente é necessário analisar a métrica convencional e compreender suas limitações. Partindo deste princípio, buscou-se aplicar o EVM para estimar a BER em sistemas de comunicações ópticas coerentes digitais de um único *span* operando em diversas condições.

Para isso, considerou-se um sistema óptico passivo de 150 km de modo que, por se tratar de um enlace relativamente longo para os moldes de uma PON, o sistema mostra-se limitado por ruído aditivo gaussiano para uma larga faixa de valores de potência de transmissão. Computaram-se então os valores de BER para cada LOP por meio de contagem de erros, e compararam-se os resultados obtidos aos estimados por EVM.

Particularmente, a estimativa de BER por EVM foi calculada baseada na metodologia descrita na Seção 2.3, computando-se a potência do ruído por meio do cálculo da variância dos símbolos das constelações recebidas. A comparação entre os valores estimados por contagem de erros e por EVM pode ser analisada na Figura 22.

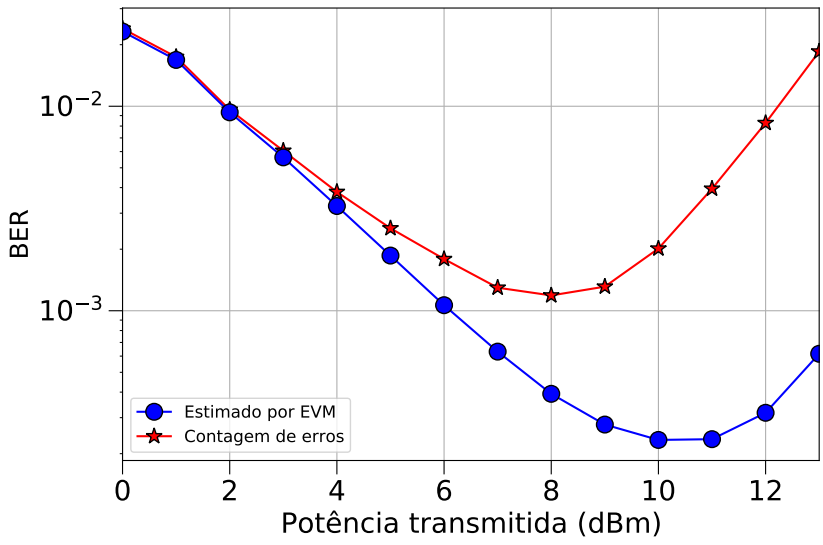


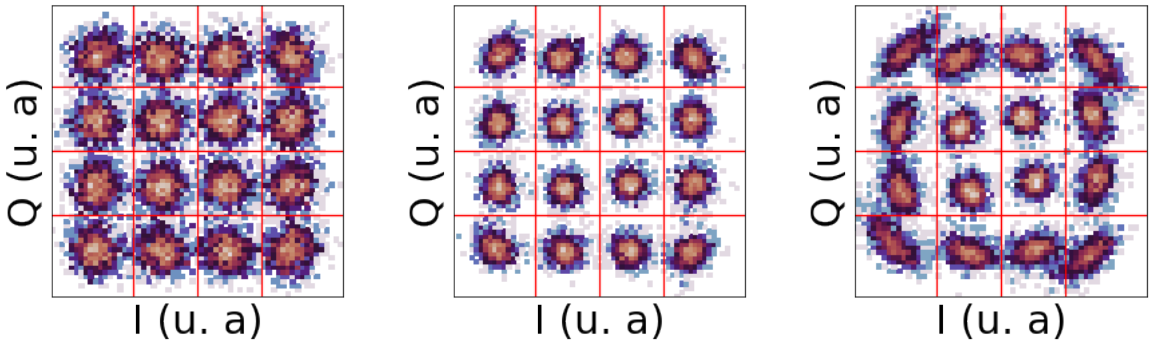
Figura 22 – BER obtida por meio de contagem de erros e estimada por EVM em função da potência de transmissão para um enlace óptico passivo de 150 km.

O gráfico comparativo evidencia que a BER predita para valores de potência de transmissão inferiores a 4 dBm são de fato muito próximos aos valores estimados por contagem de erros, o que nos mostra que, pelo menos para esta faixa de LOP, o EVM é uma métrica válida de estimativa. Nesta faixa, o aumento de potência resulta na diminuição da BER, que pode ser justificada pelo aumento da SNR do sinal. É perceptível que para estes valores de LOP o sistema é limitado por ruído gaussiano aditivo.

Conforme aumenta-se a potência de transmissão, a modulação de fase não linear passa a ser cada vez mais significativa, de modo que as distribuições de alguns símbolos da constelação deixam de ser representadas puramente por uma função densidade de probabilidade gaussiana com simetria circular. Sendo assim, o método do EVM para estimativa da BER deixa de ser válida, uma vez que ele deixa de considerar a real distribuição de probabilidade dos símbolos distorcidos.

Para que se possa comprovar esta hipótese, é necessário visualizar as constelações recebidas neste enlace para diferentes valores de potência. A Figura 23 apresenta os histogramas destas constelações considerado valores de LOP de 2, 8 e 12 dBm, bem como as regiões de decisão para detecção por máxima verossimilhança (limitadas pelas linhas vermelhas) considerando a ocorrência de ruído gaussiano aditivo.

É possível notar que, para um LOP de 2 dBm, o sistema está limitado por ruído, uma vez que a modulação de fase não linear devido ao efeito Kerr é imperceptível. Especificamente para este caso, as regiões de decisão para detecção por máxima verossimilhança mostram-se eficientes para detectar os símbolos com a menor BER possível.



(a) (b) (c)

Figura 23 – Histogramas das constelações recebidas em um sistema óptico passivo de um único canal de 150 km de comprimento para potência transmitida de (a) 2, (b) 8 e (c) 12 dBm e regiões de decisão para detecção por máxima verossimilhança. Valores de amplitude em unidades arbitrárias.

Considerando uma potência de transmissão de 8 dBm, é perceptível que a melhoria na SNR do sinal é alcançada em detrimento de uma modulação de fase não linear mais significativa. Para este caso, embora os símbolos menos energéticos tenham sido pouco afetados pelo efeito Kerr, aqueles com maior amplitude sofreram uma significativa distorção e consequentemente deixaram de apresentar uma distribuição de probabilidade gaussiana de variância constante na direção radial. Esta distorção torna-se ainda mais evidente para o caso em que considera-se o LOP de 12 dBm, pois a constelação é completamente distorcida, inviabilizando o método de detecção convencional (máxima verossimilhança).

Sendo assim, as discrepâncias entre os valores de BER estimados por EVM e obtidos por contagem de erros em sistemas ópticos digitais coerentes limitados por modulação de fase não linear justificam a necessidade de se desenvolver e aplicar técnicas mais sofisticadas para estimar a BER. Em particular, a seção seguinte apresentará como alternativa a predição por meio do processamento de constelações utilizando CNNs.

5.2 Análise da probabilidade de erro de bit estimada por meio de redes neurais convolucionais

Com a finalidade de validar o funcionamento de CNNs para predição da BER, implementou-se uma CNN MobileNetV2, assim como especificado na Seção 4.2, para processar as constelações de sinais. Neste cenário, um bloco de dados de 6000 histogramas de constelações com 56 *bins*, cada uma gerada por 10.000 símbolos, foi dividido em grupos de treino e validação cruzada, assim como já especificado. É válido relembrar que as imagens consideram sistemas de com comprimentos que variam de 80 a 150 km e valores de potência de 0 a 13 dBm para assegurar uma boa capacidade de generalização da rede.

Após o tratamento dos dados, a CNN foi submetida a um treinamento performado em 300 iterações, e o erro quadrático médio, definido como função de custo do algoritmo, foi levantado ao longo do processo para os dados de treinamento e validação cruzada. A evolução da função de custo durante o processo de treinamento pode ser analisada na Figura 24.

Ao observar o comportamento do erro quadrático médio, é possível notar que as curvas referentes aos dados de treino e validação cruzada sofrem uma drástica redução ao longo do treinamento, principalmente na etapa inicial do processo, se estabilizando em aproximadamente $7,4 \times 10^{-3}$. A partir da 70ª iteração, o custo obtido na validação

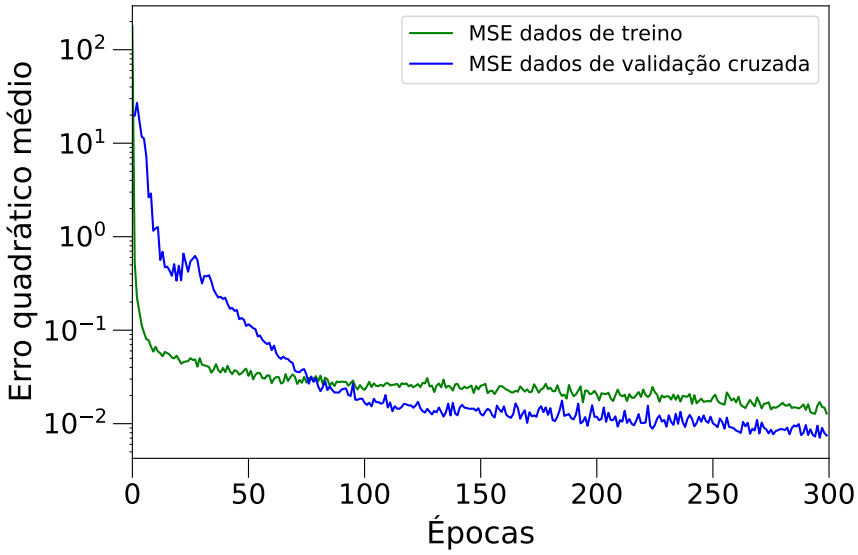


Figura 24 – Erro quadrático médio (MSE) para dados de treino e teste em função da época de treinamento de uma CNN MobileNetV2 considerando o algoritmo de treino gradiente descendente de lote.

cruzada tende a acompanhar o comportamento do custo do bloco de treino, mantendo-se abaixo dele durante todo o restante do processo. Sendo assim, é possível notar a ausência de *overffiting*, uma vez que a rede mostrou-se capaz de obter baixo custo mesmo em dados não utilizados no processo de otimização do custo.

Após treinar a CNN, realizou-se um teste de predição da BER para os histogramas contidos no bloco de validação cruzada. Neste procedimento, foram comparados os valores de BER estimados por contagem de erros, utilizados para rotular as imagens, e os valores preditos pelo algoritmo. O resultado pode ser analisado na Figura 25.

Ao se comparar os valores estimados por CNN com os rótulos, é perceptível que existe um alto grau de proximidade entre eles. Nota-se que as saídas preditas encontram-se espalhadas em torno a curva ideal (BER estimada por contagem = BER estimada por CNN), evidenciando assim a capacidade de generalização da CNN para prever os valores de BER nos mais diferentes cenários.

Sabe-se que a curva ideal, representada pela reta bissetriz, deve idealmente contemplar toda a variância das amostras, uma vez que o espalhamento na direção ortogonal a ela está diretamente associado ao erro de estimativa. Sendo assim, a análise da variância nesta direção por meio da aplicação do PCA (Seção 3.3) nos possibilita mensurar quantitativamente o percentual da variância total que está associado ao erro.

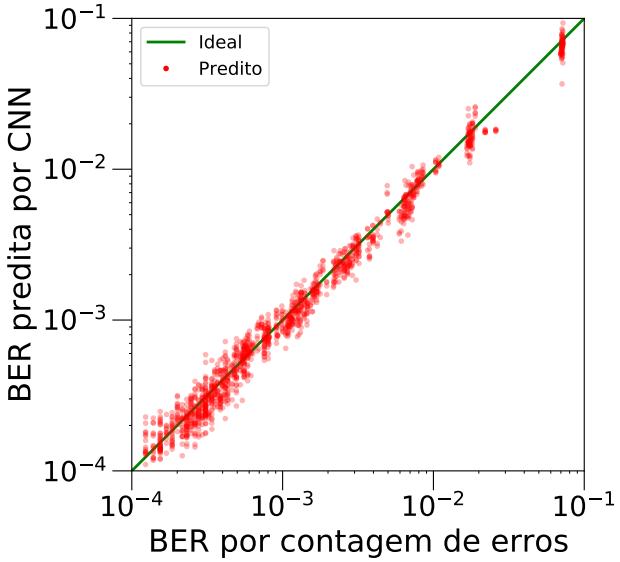


Figura 25 – BER obtida por meio de regressão de CNN em função da BER obtida por meio de contagem de erros. Dados são referentes ao processamento de histogramas formados por 10.000 símbolos e 56 *bins* em um conjunto de teste após o processo de treinamento da CNN.

Tabela 3 – Autovetores do bloco de dados de validação cruzada que relaciona BER predita por CNN aos valores rotulados para *dataset* de histogramas gerados por 10.000 símbolos e 56 *bins*.

Autovetores	
$\vec{e}_1$	(0.710942, 0.703250)
$\vec{e}_2$	(-0.703250, 0.710942)

As Tabelas 3 e 4 apresentam os autovetores e autovalores obtidos considerando os dados da Figura 25. Um ponto importante a se destacar é que a direção apontada pelo autovetor $\vec{e}_1$ apresenta um ângulo de 44.69° com relação ao eixo das abscissas, valor muito próximo à inclinação da bissetriz, que é de 45°. A variância nessa direção equivale a 99.63% da variância total do sistema, o que nos mostra que o percentual associado à direção de erro ($\vec{e}_2$) é de apenas 0.37%. Sendo assim, a CNN se mostra capaz não somente de estimar a BER para os mais diferentes cenários, mas também realizar esta tarefa com um alto grau de precisão.

Ao comprovar o funcionamento do sistema preditivo, faz-se necessário analisar o desempenho das CNNs considerando histogramas com diferentes quantidades de símbolos, uma vez que a redução deste parâmetro é de extrema importância para a redução da latência no processo de predição. Este procedimento será descrito na seção seguinte.

Tabela 4 – Autovalores do bloco de dados de validação cruzada que relaciona BER predita por CNN aos valores rotulados para *dataset* de histogramas gerados por 10.000 símbolos e 56 *bins*.

Direção	$\vec{e}_1$	$\vec{e}_2$
Autovalor	0.948618	0.003513
% na variância	99.63	0.37

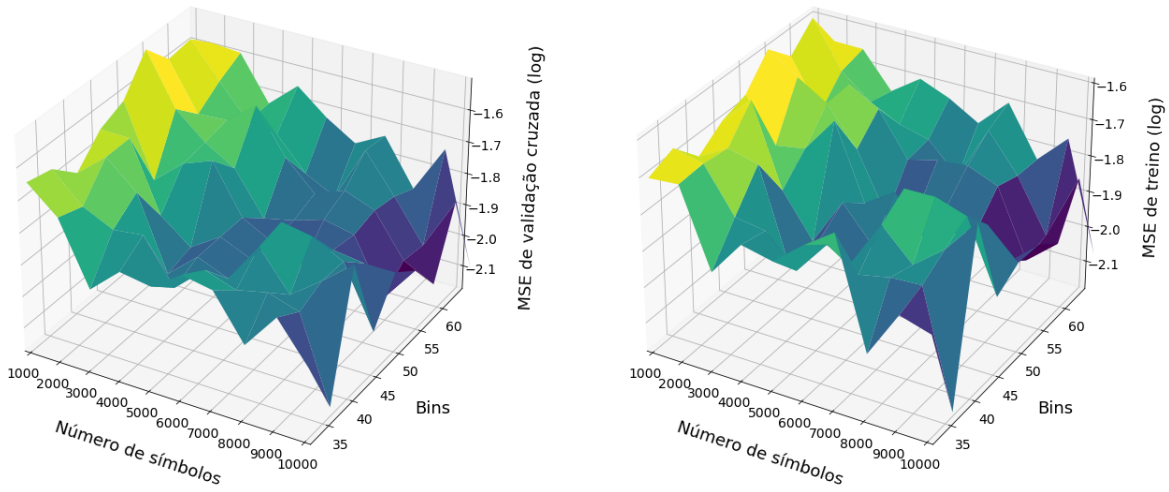
5.3 Análise do desempenho das redes neurais convolucionais para diferentes cenários

Para viabilizar a análise do desempenho das CNNs para diversas configurações de *datasets*, repetiu-se o processo de treinamento e validação descrito na Seção 5.2 considerando blocos de dados formados por histogramas quadrados de constelações geradas por blocos que variaram de 1.000 a 10.000 símbolos com passo 1.000, e *bins* variando de 32 a 64 com passo 4.

Para cada uma das 90 diferentes configurações de CNN, realizou-se o treinamento em 300 épocas, calculando-se então o custo final para blocos de treino e validação cruzada. Devido à existência de fatores aleatórios que podem interferir no treinamento, tais como aleatoriedade dos pesos iniciais da rede, dos símbolos selecionados para compor cada histograma e da escolha de amostras para compor os blocos de treino, validação cruzada e teste, repetiu-se 10 vezes o treinamento para cada uma das 90 configurações.

A finalidade de se performar múltiplos treinamentos para cada *dataset* é a de suavizar a superfície de MSE, considerando sempre o valor médio entre os diferentes resultados obtidos para cada caso. Idealmente este procedimento poderia considerar uma quantidade maior de treinamentos por configuração (não apenas 10), porém este valor foi definido com base nas limitações de recursos computacionais para performar os treinamentos, uma vez que os algoritmos apresentam alto nível de complexidade (vide Seção 5.5).

Após performar os treinamentos, organizaram-se graficamente os valores médios de MSE para que se pudesse analisar o desempenho das redes considerando as diferentes situações descritas. Estes valores, bem como os de variância, também foram dispostos em tabelas para facilitar a análise dos dados. Os dados obtidos podem ser analisados na Figura 26 e nas Tabelas de 5 a 8.



(a) (b)

Figura 26 – Erro quadrático médio para diferentes valores de resolução e quantidades de símbolo referentes aos conjuntos de (a) validação cruzada e (b) treino.

Ao se observar o valor médio de MSE para treino e validação cruzada em cada uma das diferentes configurações, é possível notar inicialmente um comportamento pouco suave das superfícies, o que possivelmente está relacionado com os fatores aleatórios já mencionados. Além disso, uma tendência perceptível é a redução do MSE a medida que se aumenta o número de símbolos.

Tabela 5 – Valor médio de MSE de treino considerando o resultado de 10 treinamentos para cada um dos 90 *datasets* em escala logatirmica (log_{10})

	Número de simbolos por histograma									
Bins	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
32	-1.703	-1.696	-1.910	-1.810	-1.828	-1.761	-1.702	-2.002	-1.870	-1.895
36	-1.674	-1.692	-1.814	-1.826	-1.911	-1.814	-1.870	-1.852	-1.847	-2.168
40	-1.747	-1.702	-1.722	-1.985	-1.879	-1.920	-1.790	-1.874	-1.807	-1.709
44	-1.699	-1.621	-1.845	-1.963	-1.831	-1.927	-1.745	-1.744	-1.771	-1.967
48	-1.676	-1.795	-1.781	-1.681	-1.886	-1.832	-1.939	-1.896	-1.848	-1.929
52	-1.599	-1.654	-1.705	-1.810	-1.973	-1.944	-1.901	-1.940	-2.008	-1.934
56	-1.664	-1.801	-1.701	-1.746	-1.867	-1.845	-1.837	-1.872	-2.058	-1.985
60	-1.602	-1.731	-1.798	-1.839	-1.860	-1.943	-1.788	-1.894	-2.040	-1.854
64	-1.714	-1.715	-1.815	-1.758	-1.791	-1.840	-1.760	-1.898	-1.816	-2.126

Tabela 6 – Valor médio de MSE de validação cruzada considerando o resultado de 10 treinamentos para cada um dos 90 *datasets* em escala logatirmica (log_{10}

	Número de simbolos por histograma									
Bins	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
32	-1.647	-1.727	-1.922	-1.838	-1.854	-1.789	-1.807	-1.945	-1.830	-1.954
36	-1.665	-1.754	-1.798	-1.787	-1.922	-1.838	-1.870	-1.849	-1.879	-2.146
40	-1.736	-1.659	-1.743	-1.971	-1.889	-1.928	-1.800	-1.920	-1.848	-1.768
) 44	-1.656	-1.658	-1.875	-1.949	-1.887	-1.928	-1.784	-1.802	-1.852	-2.043
48	-1.622	-1.848	-1.770	-1.734	-1.887	-1.939	-1.984	-1.906	-1.920	-1.920
52	-1.515	-1.678	-1.759	-1.816	-1.950	-1.929	-1.899	-1.920	-2.097	-1.978
56	-1.599	-1.750	-1.744	-1.673	-1.812	-1.903	-1.865	-1.902	-2.037	-2.086
60	-1.587	-1.698	-1.805	-1.820	-1.923	-1.972	-1.795	-1.917	-2.049	-1.883
64	-1.652	-1.660	-1.801	-1.719	-1.796	-1.855	-1.805	-1.951	-1.796	-2.162

Tabela 7 – Variância de MSE de treino considerando o resultado de 10 treinamentos para cada um dos 90 *datasets*

	Número de símbolos por histograma									
Bins	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
32	1.7e-5	1.5e-5	5.0e-5	2.6e-5	2.4e-5	3.1e-5	7.4e-5	8.5e-5	4.3e-5	7.9e-5
36	1.3e-5	2.2e-5	2.2e-5	5.2e-5	5.3e-5	3.9e-5	5.5e-5	5.7e-5	3.5e-5	1.4e-4
40	5.7e-6	6.1e-5	2.5e-5	7.6e-5	4.0e-5	6.2e-5	2.2e-5	5.2e-5	4.7e-5	1.9e-5
44	1.4e-6	5.1e-5	4.1e-5	6.6e-5	3.2e-5	6.1e-5	1.3e-5	2.6e-5	3.9e-5	6.8e-5
48	5.8e-5	2.3e-5	1.5e-5	2.0e-5	7.3e-5	9.2e-5	5.9e-5	4.1e-5	3.2e-5	6.0e-5
52	3.9e-5	3.4e-5	2.1e-6	3.3e-5	8.5e-5	6.1e-5	4.8e-5	7.3e-5	8.4e-5	5.9e-5
56	9.4e-6	1.4e-5	7.6e-6	7.9e-6	4.5e-5	5.0e-5	5.2e-5	3.8e-5	9.9e-5	7.7e-5
60	7.3e-5	1.2e-5	1.6e-5	3.2e-5	8.6e-5	6.6e-5	5.7e-5	8.8e-5	1.0e-4	3.5e-5
64	2.4e-6	1.0e-5	2.2e-5	5.4e-5	5.1e-5	3.4e-5	2.7e-5	5.5e-5	3.1e-5	1.3e-4

Tabela 8 – Variância de MSE de validação cruzada considerando o resultado de 10 treinamentos para cada um dos 90 *datasets*

	Número de símbolos por histograma									
Bins	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
32	1.4e-5	1.9e-5	9.3e-5	5.7e-5	6.9e-5	5.3e-5	6.5e-5	1.1e-4	6.8e-5	1.3e-4
36	7.1e-6	2.3e-5	4.2e-5	6.2e-5	9.3e-5	6.9e-5	8.5e-5	8.4e-5	8.6e-5	1.9e-4
40	1.6e-5	7.0e-5	3.4e-5	1.2e-4	7.6e-5	1.0e-4	4.5e-5	9.3e-5	7.3e-5	3.1e-5
44	3.1e-6	3.2e-5	7.9e-5	1.1e-4	8.2e-5	1.0e-4	5.0e-5	5.3e-5	7.5e-5	1.5e-4
48	2.3e-5	5.8e-5	3.1e-5	2.6e-5	8.7e-5	1.2e-4	1.2e-4	8.4e-5	9.1e-5	9.2e-5
52	5.4e-5	1.0e-5	3.3e-5	5.1e-5	1.2e-4	9.7e-5	8.5e-5	9.8e-5	1.7e-4	1.2e-4
56	8.3e-6	2.3e-5	2.4e-5	3.2e-5	5.3e-5	8.6e-5	9.9e-5	8.7e-5	1.4e-4	1.7e-4
60	2.1e-5	1.1e-5	3.8e-5	5.2e-5	1.1e-4	1.2e-4	7.0e-5	1.3e-4	1.6e-4	7.9e-5
64	6.2e-6	4.7e-6	4.9e-5	7.3e-5	6.4e-5	6.5e-5	5.6e-5	1.1e-4	6.8e-5	2.0e-4

Embora o comportamento das curvas não seja suave, nota-se que, de modo geral, todos os casos convergiram para valores de custo de treino (MSE_{treino}) e de validação cruzada ($MSE_{Xvalidation}$) inferiores a $2,52 \times 10^{-2}$ e $3,06 \times 10^{-2}$, respectivamente. É perceptível que, considerando as faixas adotadas, conforme se aumenta a quantidade de símbolos dos histogramas que compõem os blocos de dados, os valores de MSE tendem a ser cada vez menores.

Analisando os valores de variância computados por meio dos resultados de 10 treinamentos para cada um dos 90 diferentes *datasets*, é possível notar que, de modo geral, todos os valores são da ordem de 10^{-4} ou inferiores, tanto para teste quanto para validação. Considerando que os valores de MSE para teste e validação estão contidos nos intervalos de $[6,79 \times 10^{-3}, 2,52 \times 10^{-2}]$ e $[3,06 \times 10^{-2}, 6,89 \times 10^{-3}]$, respectivamente, todos os valores de variância são pelo menos 10 vezes inferiores aos valores médios obtidos.

Dentre os resultados obtidos, a configuração que obteve o menor custo ao final de validação cruzada foi a que utilizou histogramas de constelações formadas por 10.000 símbolos e 64 *bins*, apresentando MSE de $7,49 \times 10^{-3}$ e $6,89 \times 10^{-3}$ para treino e validação, respectivamente.

Sendo assim, é possível adotar esta configuração como referência para um comparativo de desempenho, validando assim este modelo preditivo por meio da comparação com os valores de BER estimados por EVM e obtidos por meio de contagem de erros.

5.4 Comparativo entre desempenho do EVM e CNNs para predição de BER

Sabe-se que devido a natureza aleatória de sistemas de comunicação, o sinal detectado no receptor dificilmente gerará constelações de sinais absolutamente idênticas. Sendo assim, para possibilitar uma comparação justa entre resultados obtidos por EVM e CNN, utilizaram-se os dados simulados do enlace óptico passivo de 150 km, já discutido na Seção 5.1.

Geraram-se então 1.000 histogramas de 64 *bins* de constelações compostas por 10.000 símbolos selecionados aleatoriamente, para cada um dos 14 diferentes valores de potência. Essas constelações foram processadas pela CNN e geraram, para cada LOP, 1.000 estimativas de BER. É importante ressaltar que a CNN escolhida para processar os dados foi o modelo que apresentou o pior MSE de validação cruzada para o *dataset* em questão (histogramas de 64 *bins* e 10000 símbolos) dentre as 10 redes treinadas. Esta é uma medida tomada visando assegurar que o desempenho médio seja, pelo menos na maior parte dos casos, superior ao apresentado a seguir. A Figura 27 e a Tabela 9 apresentam os histogramas obtidos para cada valor de potência e a comparação entre BER estimada por contagem de erros e valor médio estimado por CNN, respectivamente.

Ao se observar os gráficos, é possível perceber que para alguns casos as distribuições de probabilidades apresentam comportamento semelhante ao de uma distribuição gaussiana, porém esta afirmação não é válida para todos os histogramas. Na maior parte dos casos, o valor de BER obtido por meio da contagem de erros é próximo ao valor médio estimado pela rede, normalmente contando com um pequeno *offset*. A variância máxima obtida ocorre para potência de 13 dBm ($6,56 \times 10^{-6}$), e a mínima está associada ao LOP de 7 dBm ($2,58 \times 10^{-8}$).

Ao se comparar os valores de BER estimados por contagem de erros aos valores médios estimados pela CNN, nota-se que a maior divergência ocorre para a potência de 3 dBm, cujo erro percentual é dado por 15,83%. Estritamente para a o valor de LOP de 11 dBm, o erro atinge o valor mínimo de 0,66%.

Destaca-se ainda no gráfico as regiões 4 sigma, sendo sigma o desvio padrão de cada uma das distribuições. Considerando as distribuições de probabilidade da Figura 27, esta região abrange pelo menos 94,3% de todas as estimativas. Deste modo, é possível afirmar que a maior parte das predições ocorrerão dentro desta faixa de valores. Tais regiões são

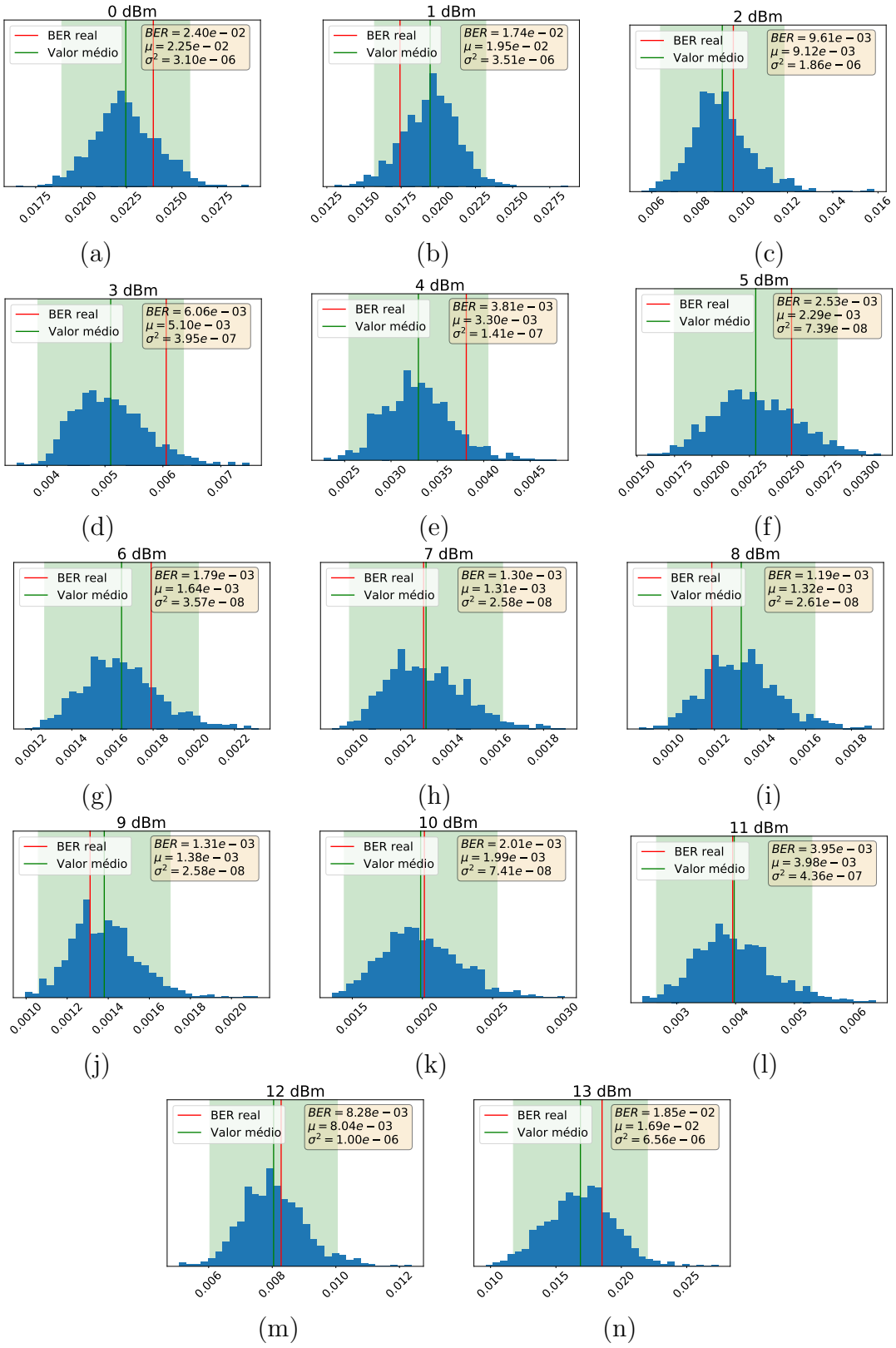


Figura 27 – Histogramas de regressão de estimativa de BER preditos por CNN treinada com histogramas de 10000 símbolos e 64 *bins* para sistema óptico passivo de 150 km e valores de potência de transmissão de (a) 0, (b) 1, (b) 2, (b) 3, (b) 4, (b) 5, (b) 6, (b) 7, (b) 8, (b) 9, (b) 10, (b) 11, (b) 12 e (b) 13 dBm. Regiões 4 sigma destacadas em verde.

Tabela 9 – Comparação entre valores de BER estimados por contagem de erros e valores médios obtidos por CNN (10.000 símbolos - 64 *bins*)

Potência (dBM)	BER - Contagem de erros	BER - Média CNN	Erro (%)
0	2.40e-02	2.25e-02	6.32
1	1.74e-02	1.95e-02	11.64
2	9.61e-03	9.12e-03	5.07
3	6.06e-03	5.10e-03	15.83
4	3.81e-03	3.30e-03	13.55
5	2.53e-03	2.29e-03	9.45
6	1.79e-03	1.64e-03	8.12
7	1.30e-03	1.31e-03	0.79
8	1.19e-03	1.32e-03	10.87
9	1.31e-03	1.38e-03	5.26
10	2.01e-03	1.99e-03	1.32
11	3.95e-03	3.98e-03	0.66
12	8.28e-03	8.04e-03	2.86
13	1.85e-02	1.69e-02	8.93

relevantes para a comparação entre os resultados obtidos por meio de EVM e CNN, assim como apresentado na Figura 28.

Percebe-se que, considerando os valores médios preditos por CNN, a curva de BER é extremamente próxima à do valor estimado por contagem de erros, o que nem sempre acontece se levarmos em consideração a curva obtida por EVM. Um fator de destaque na predição por CNN é o fato de que ela é valida para toda a faixa de potência, sendo eficiente para estimar os valores de BER para sistemas tanto limitados por ruído gaussiano quanto por modulação de fase não linear. Trata-se portanto de uma rede neural com alta capacidade de generalização, que pode ser aplicada para sistemas com diferentes valores de LOP e comprimentos de enlace.

Embora os resultados obtidos para a CNN considerando a configuração de histogramas com 10.000 símbolos e 64 *bins* sejam satisfatórios, é necessário lembrar que a redução na quantidade de símbolos é um importante fator a ser considerado. Sendo assim, aplicou-se o mesmo tipo de análise para uma CNN considerando histogramas gerados por 1.000 símbolos. Levando em consideração o processo de treinamento previamente descrito, o caso

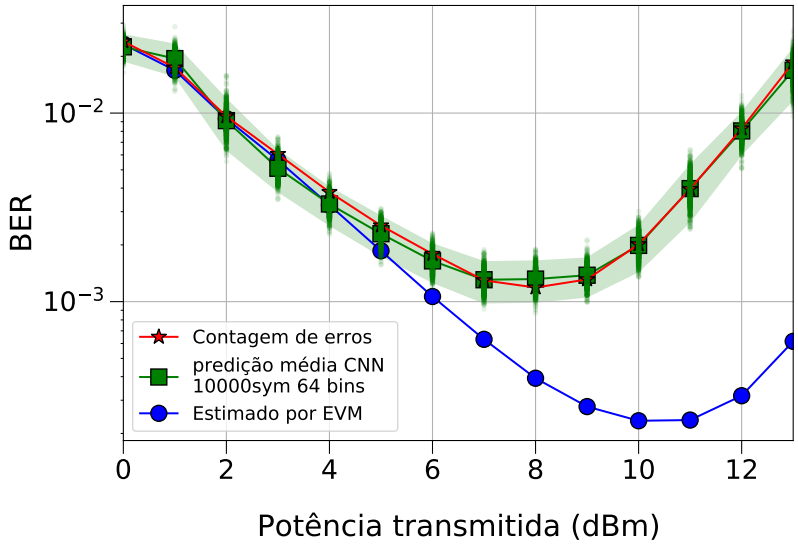


Figura 28 – Taxa de erro de bit obtida por meio de contagem de erros, predita por EVM e por CNN (histogramas de 10.000 símbolos de 64 *bins*) em função da potência de transmissão para enlace óptico passivo de 150 km.

que obteve o melhor desempenho seguindo esta limitação é o que considerou histogramas de 40 *bins*. A seguir, a Figura 29 apresenta o gráfico de BER em função de LOP para este caso. Os histogramas para cada potência podem ser analisados na Figura 30.

Ao se observar os histogramas da Figura 30, é possível perceber que, de modo geral, as variâncias obtidas são consideravelmente maiores (da ordem de grandeza de 10^{-5} ou inferior) se comparadas à configuração anterior. Assim como no outro caso, para a

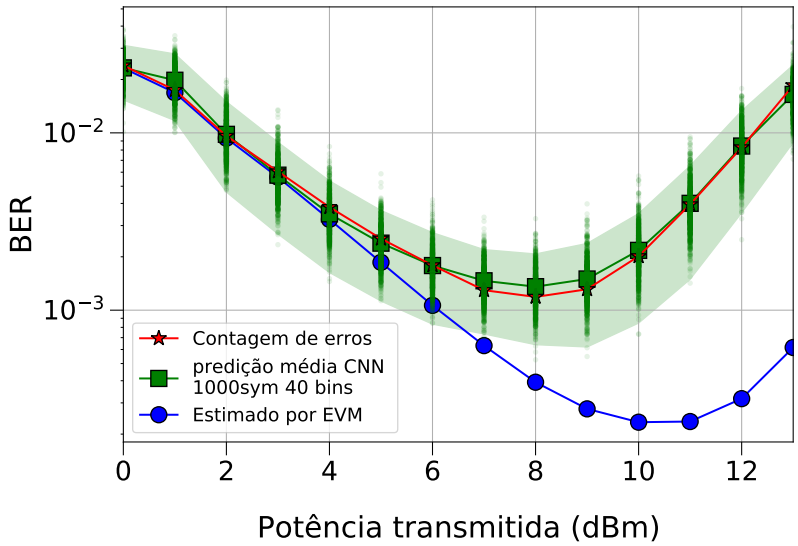


Figura 29 – Taxa de erro de bit obtida por meio de contagem de erros, predita por EVM e por CNN (histogramas de 1.000 símbolos de 40 *bins*) em função da potência de transmissão para enlace óptico passivo de 150 km.

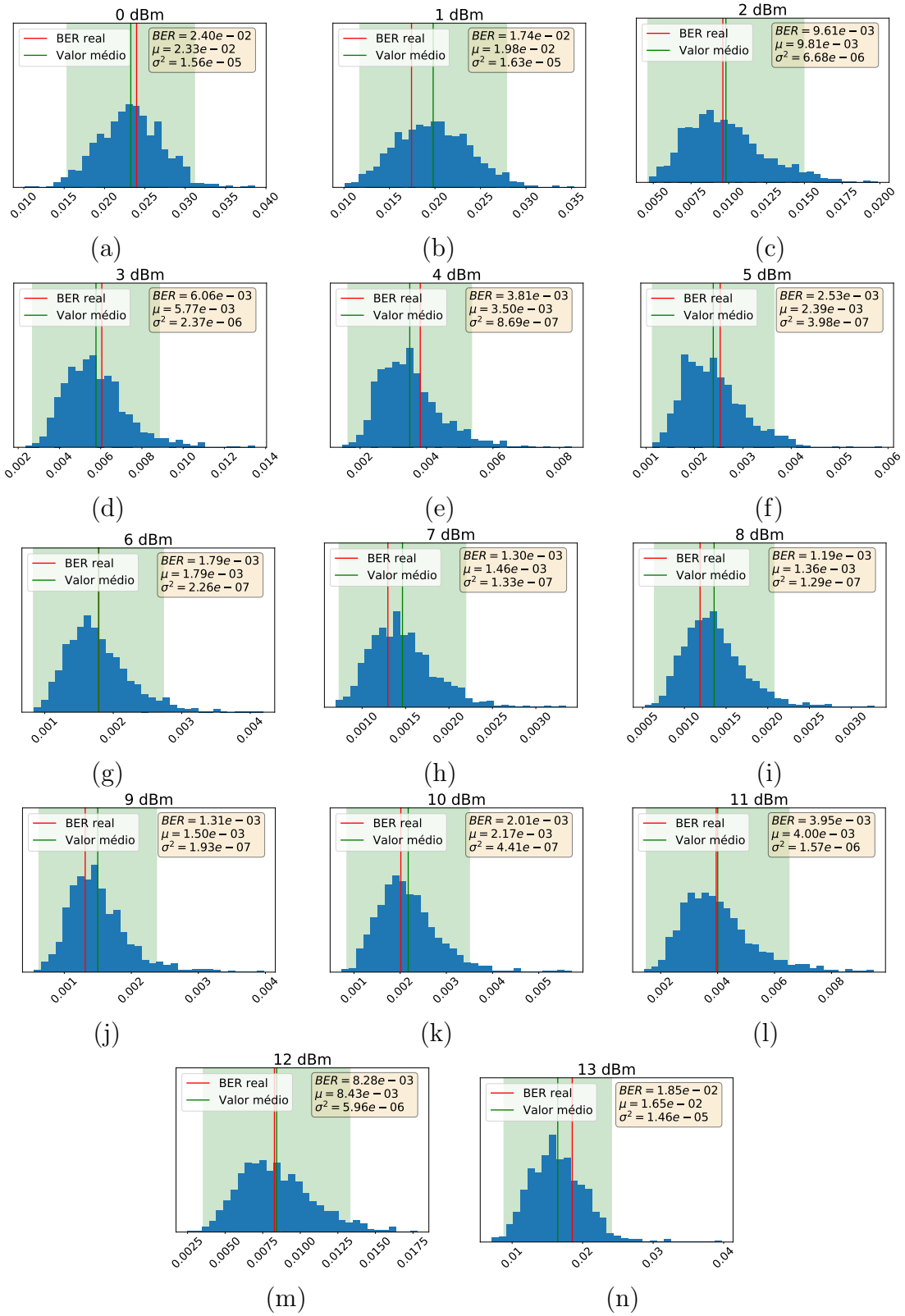


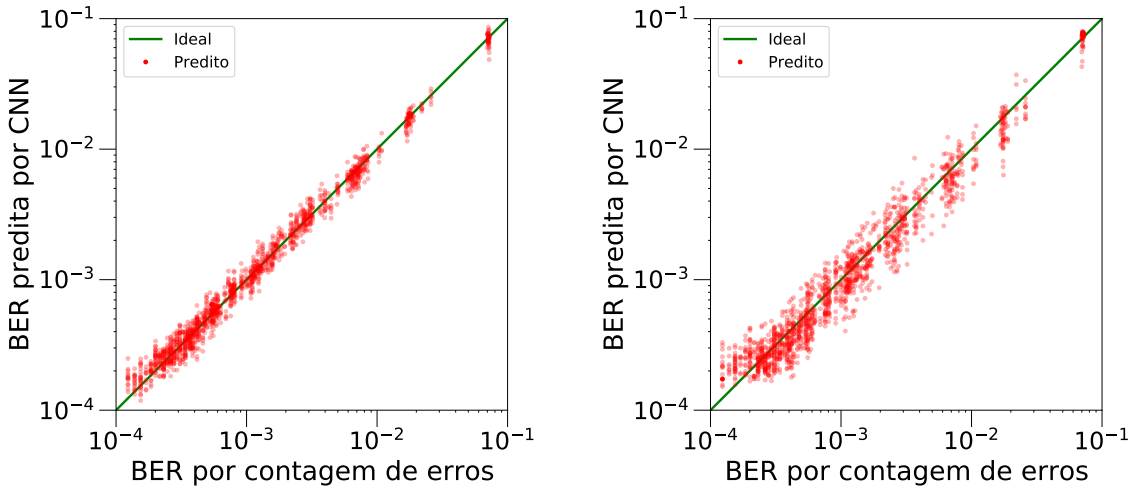
Figura 30 – Histogramas de regressão de estimativa de BER preditos por CNN treinada com histogramas de 1.000 símbolos com 40 *bins* para sistema óptico passivo de 150 km e valores de potência de transmissão de (a) 0, (b) 1, (b) 2, (b) 3, (b) 4, (b) 5, (b) 6, (b) 7, (b) 8, (b) 9, (b) 10, (b) 11, (b) 12 e (b) 13 dBm. Regiões 4 sigma destacadas em verde.

Tabela 10 – Comparação entre valores de BER estimados por contagem de erros e valores médios obtidos por CNN (1.000 símbolos - 40 *bins*)

Potência (dBM)	BER - Contagem de erros	BER - Média CNN	Erro (%)
0	2.40E-02	2.33E-02	3.01
1	1.74E-02	1.98E-02	13.65
2	9.61E-03	9.81E-03	2.10
3	6.06E-03	5.77E-03	4.69
4	3.81E-03	3.50E-03	8.30
5	2.53E-03	2.39E-03	5.65
6	1.79E-03	1.79E-03	0.28
7	1.30E-03	1.46E-03	12.98
8	1.19E-03	1.36E-03	14.17
9	1.31E-03	1.50E-03	14.28
10	2.01E-03	2.17E-03	7.95
11	3.95E-03	4.00E-03	1.35
12	8.28E-03	8.43E-03	1.79
13	1.85E-02	1.65E-02	11.09

maior parte dos valores de LOP, o valor de BER obtido por meio da contagem de erros é próximo ao valor médio estimado pela rede, normalmente contando com um pequeno *offset*. Particularmente para este caso, a maior divergência entre a média estimada por CNN e por BER predita por contagem de erros ocorreu para a potência de 9 dBm e foi equivalente a um desvio de aproximadamente 14,28%.

Percebe-se que, considerando os valores médios preditos por CNN, a curva de BER é novamente próxima à do valor real, todavia o aumento na variância gera uma maior imprecisão nos valores estimados. Tal particularidade pode inclusive ser observada analisando o aumento das fronteiras da região 4 sigma. Embora o desempenho desta CNN seja relativamente inferior ao do caso anterior, ela ainda se mostra capaz de estimar os valores de BER para sistemas tanto limitados por ruído gaussiano quanto por modulação de fase não linear, se tratando de uma alternativa mais atraente que o EVM para casos em que o LOP é superior a 7 dBm. Novamente a rede neural apresenta alta capacidade de generalização, e pode ser aplicada para sistemas com diferentes valores de LOP e



(a) (b)

Figura 31 – BER obtida por meio de regressão de CNN em função da BER obtida por meio de contagem de erros. Dados referentes às redes treinadas considerando histogramas gerados por (a) 10.000 símbolos - 64 *bins* e (b) 1.000 símbolos - 40 *bins*.

Tabela 11 – Autovetores do bloco de dados de validação cruzada que relaciona BER predita por CNN aos valores rotulados para *dataset* de histogramas gerados por 10.000 símbolos - 64 *bins* e 1.000 símbolos - 40 *bins*.

Autovetores	10.000 sym - 64 bins	1.000 sym - 40 bins
$\vec{e}_1$	(0.713536, 0.700618)	(0.714038, 0.700107)
$\vec{e}_2$	(-0.700618, 0.713536)	(-0.700107, 0.714038)

comprimentos de enlace, porém, com precisão reduzida se comparada à configuração previamente discutida.

Considerando as diferenças de desempenho obtidas no procedimento de teste para as duas configurações de CNN, é possível aplicar critérios quantitativos para avaliar por meio do PCA a variância relativa ao erro de predição, assim como realizado na seção anterior. Para isso, pode-se novamente representar os dados de predição de BER utilizando CNNs em função dos valores rotulados. A Figura 31 apresenta estes dados considerando o conjunto de validação cruzada dos *datasets* compostos por histogramas de 10.000 símbolos - 64 *bins* e 1.000 símbolos - 40 *bins*.

Considerando os conjuntos de dados em questão, calcularam-se os autovalores e autovetores para possibilitar a análise de dispersão de dados para ambos os casos. Os resultados obtidos podem ser analisados nas Tabelas 11 e 12.

Tabela 12 – Autovalores do bloco de dados de validação cruzada que relaciona BER predita por CNN aos valores rotulados para *dataset* de histogramas gerados por 10.000 símbolos - 64 *bins* e 1.000 símbolos - 40 *bins*.

Autovalores	10.000 sym - 64 bins	1.000 sym - 40 bins
λ_1	0.825424	0.808048
λ_2	0.002488	0.00881
% na variância de λ_1	99.70	98.92
% na variância de λ_2	0.30	1.08

Nota-se que para ambos os casos a direção de maior dispersão, regida pelo autovalor $\vec{e}_1$, assume valores próximos ao da curva ideal (45° com relação ao eixo das abscissas), sendo representados pelas inclinações de $44,48°$ e $44,44°$ para os blocos de histogramas de 10.0000 e 1.000 símbolos, respectivamente. Considerando o melhor caso (10.000 símbolos - 64 *bins*), a variância associada ao erro de predição é de aproximadamente 0,30%, enquanto que o valor assumido para o outro caso é de 1,08%. Mesmo levando em conta que a parcela relativa ao erro seja baixa em ambos os casos, é necessário ressaltar que a parcela obtida para o *dataset* de histogramas de 1.000 símbolos - 40 *bins* é mais de 3,5 vezes superior à do outro caso, o que novamente corrobora para o aumento da variância, e consequente queda de desempenho durante a predição dos dados.

Embora haja uma diferença perceptível entre os dois casos apresentados, a regressão por meio do processamento de histogramas compostos por 1.000 símbolos pela CNN pode ainda assim estimar de forma suficientemente precisa probabilidades de erro de bit inferiores a 10^{-3}, o que não pode ser feito por meio de contagem de erros (neste caso o menor valor de BER que se pode estimar é justamente 10^{-3}). Sendo assim, a CNN proposta permite quantificar a qualidade do sinal utilizando janelas de observação mais curtas do que as necessárias para performar contagem de erros, o que possibilita um controle com menor latência.

5.5 Análise de complexidade

Após treinar e validar os resultados, fez-se necessário computar a quantidade de operações de ponto flutuante necessárias para se estimar a BER em um sistema de

comunicação óptica digital coerente por meio de uma CNN. Para isso, faz-se necessário analisar a estrutura do algoritmo de regressão adotado.

Assim como apresentado previamente, a CNN MobileNetV2 é composta originalmente por um total de 3.538.984 parâmetros, dos quais 3.504.872 são treináveis e 34.112 não treináveis. Esses parâmetros compõem uma rede neural de classificação que recebe em sua entrada um tensor de dimensão (224, 224, 3), e gera uma saída composta por um vetor unidimensional de 1000 elementos (classes).

É necessário considerar que a arquitetura adotada implementa alterações na estrutura original da rede, especificamente nas camadas de entrada e saída. Para o presente problema, considerou-se um tensor de entrada com dimensão (128, 128, 3) e apenas uma saída, uma vez que a tarefa performada deixou de ser classificação e passou a ser regressão. Sendo assim, a quantidade total de parâmetros foi reduzida para um total de 2.259.265, dos quais 2.225.153 são treináveis e 34.112 são não treináveis.

A redução da quantidade de parâmetros da rede gerou consequentemente uma redução na quantidade total de operações de ponto flutuante. Embora a topologia original demande um total de aproximadamente 601,61 MFlop para processar um tensor, este valor foi reduzido para 195,61 MFlop.

Portanto, o custo computacional necessário para estimar a BER por meio do processamento de uma constelação de um sinais considerando um sistema óptico coerente é de 195.611.142 operações de ponto flutuante. É válido destacar que grande parte das operações podem ser implementadas de forma paralela, todavia, a implementação eficiente do algoritmo pode ser tópico de um trabalho futuro.

6 Conclusões

Com base nos resultados apresentados, é possível concluir que o EVM, convencionalmente utilizado para estimativa de BER em sistemas limitados por ruído gaussiano aditivo, possui restrições quando aplicado a constelações de sistemas de comunicações ópticas digitais coerentes sob efeito significativo de modulação de fase não linear. Neste tipo de ocasião, torna-se perceptível o baixo desempenho desta metodologia para a predição de BER, uma vez que ela não considera as reais distribuições de probabilidade dos diferentes símbolos que compõem a constelação de sinais.

Em contrapartida, a predição baseada no processamento de histogramas de constelações utilizando CNNs apresentou resultados satisfatórios para realizar a estimativa da BER, estando o sistema limitado ou não por não linearidades. Os resultados mostram que é possível aplicar CNNs para estimar a BER considerando histogramas de constelações constituídas por diferentes quantidades de símbolos. O melhor resultado obtido, que considerou histogramas compostos por conjuntos de 10.000 símbolos e 64 *bins*, gerou uma estimativa média de BER com erro inferior a 16% para uma rede PON de 150 km levando em consideração os diferentes valores de potência de transmissão. Nesta ocasião, foi possível obter erro inferior a 1% para potência de transmissão de 7 e 11 dBm.

Embora o melhor desempenho na estimativa tenha sido performado considerando histogramas formados por 10.000 símbolos, os resultados mostram que a predição de BER pode ser realizada considerando sequências piloto menores. Nos casos em que se consideraram 1.000 símbolos por histograma, a precisão nos resultados foi significativamente afetada, porém o desempenho ainda se mostrou consideravelmente superior ao obtido por EVM em o sistemas limitados por modulação de fase não linear. Em todas as situações simuladas, considerando as diferentes quantidades de símbolos por histogramas e *bins*, houve convergência durante os processos de treinamento. Tornou-se evidente que algumas configurações apresentam resultados melhores que outras, porém, dentro das faixas de parâmetros considerados, todos os valores de erro quadrático médio para dados de validação cruzada foram inferiores a $3,06 \times 10^{-2}$.

Embora os resultados atestem a capacidade das CNNs para realizar a estimativa de BER em diferentes situações, a alta complexidade destes algoritmos se apresentam como sendo um considerável contra ponto a esta aplicação. Mesmo obtendo bons resultados

para sistemas ópticos limitados por ruído aditivo, este tipo de aplicação não se justifica devido a diferença entre o custo computacional do EVM e de CNNs. Todavia, as CNNs mostram-se promissoras para estimar a BER para a faixa de valores de potência em que o sistema passa a ser limitado pelo efeito Kerr óptico, uma vez que para esses casos o desempenho obtido por EVM é insatisfatório.

Referências[1]

AGRAWAL, G. P. Nonlinear fiber optics. In: *Nonlinear Science at the Dawn of the 21st Century*. [S.l.]: Springer, 2000. p. 195–211. Citado 6 vezes nas páginas 24, 25, 26, 27, 28 e 29.

AGRAWAL, G. P. *Fiber-optic communication systems*. [S.l.]: John Wiley & Sons, 2012. v. 222. Citado 8 vezes nas páginas 17, 21, 23, 24, 25, 26, 28 e 29.

AGRAWAL, G. P. Optical communication: its history and recent progress. In: *Optics in Our Time*. [S.l.]: Springer, Cham, 2016. p. 177–199. Citado na página 17.

ALEKSANDER, I.; MORTON, H. The logic of neural cognition. In: *Advanced Neural Computers*. [S.l.]: Elsevier, 1990. p. 97–102. Citado na página 31.

ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2009. Citado 11 vezes nas páginas 32, 33, 36, 37, 38, 39, 40, 43, 52, 53 e 54.

BARLETTA, L.; GIUSTI, A.; ROTTONDI, C.; TORNATORE, M. QoT estimation for unestablished lighpaths using machine learning. In: OPTICAL SOCIETY OF AMERICA. *Optical Fiber Communication Conference*. [S.l.], 2017. p. Th1J–1. Citado na página 19.

BECKER, P. M.; OLSSON, A. A.; SIMPSON, J. R. *Erbium-doped fiber amplifiers: fundamentals and technology*. [S.l.]: Elsevier, 1999. Citado 2 vezes nas páginas 18 e 25.

BRITO, J. L. S. *et al.* Pentes de frequências ópticas baseados em moduladores eletro-ópticos e fibras altamente não lineares. BDTD, 2015. Citado na página 25.

CHURCHLAND, P. S.; SEJNOWSKI, T. J. *The computational brain*. [S.l.]: MIT press, 1994. Citado na página 36.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255. Citado na página 51.

ERIKSSON, T. A.; BÜLOW, H.; LEVEN, A. Applying neural networks in optical communication systems: possible pitfalls. *IEEE Photonics Technology Letters*, IEEE, v. 29, n. 23, p. 2091–2094, 2017. Citado na página 19.

ESSIAMBRE, R.-J.; KRAMER, G.; WINZER, P. J.; FOSCHINI, G. J.; GOEBEL, B. Capacity limits of optical fiber networks. *Journal of Lightwave Technology*, IEEE, v. 28, n. 4, p. 662–701, 2010. Citado na página 18.

FATADIN, I. Estimation of BER from error vector magnitude for optical coherent systems. In: MULTIDISCIPLINARY DIGITAL PUBLISHING INSTITUTE. *Photonics*. [S.l.], 2016. v. 3, n. 2, p. 21. Citado 3 vezes nas páginas 18, 29 e 30.

FISHER, P. The pixel: a snare and a delusion. *International Journal of Remote Sensing*, Taylor & Francis, v. 18, n. 3, p. 679–685, 1997. Citado na página 44.

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

FUKUSHIMA, K.; MIYAKE, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: *Competition and cooperation in neural nets*. [S.l.]: Springer, 1982. p. 267–285. Citado na página 43.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007. Citado 8 vezes nas páginas 31, 33, 35, 37, 39, 42, 52 e 53.

HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. Citado na página 59.

JARAJREH, M. A.; GIACOUMIDIS, E.; ALDAYA, I.; LE, S. T.; TSOKANOS, A.; GHASSEMLOOY, Z.; DORAN, N. J. Artificial neural network nonlinear equalizer for coherent optical OFDM. *IEEE Photonics Technology Letters*, IEEE, v. 27, n. 4, p. 387–390, 2014. Citado na página 19.

JUNG, J. H.; SHIN, Y.; KWON, Y. Extension of convolutional neural network with general image processing kernels. In: IEEE. *TENCON 2018-2018 IEEE Region 10 Conference*. [S.l.], 2018. p. 1436–1439. Citado na página 46.

KIKUCHI, K. Fundamentals of coherent optical fiber communications. *Journal of Lightwave Technology*, IEEE, v. 34, n. 1, p. 157–179, 2015. Citado na página 17.

KOGELNIK, H. High-capacity optical communications: personal recollections. *IEEE Journal of Selected Topics in Quantum Electronics*, IEEE, v. 6, n. 6, p. 1279–1286, 2000. Citado na página 17.

KOUSHIK, J. Understanding convolutional neural networks. *arXiv preprint arXiv:1605.09081*, 2016. Citado na página 45.

LATHI, B. P. *Modern Digital and Analog Communication Systems 3e Osece*. [S.l.]: Oxford University Press, Inc., 1998. Citado na página 30.

LAVERY, D.; IONESCU, M.; MAKOVEJS, S.; TORRENGO, E.; SAVORY, S. J. A long-reach ultra-dense 10 Gbit/s wdm-pon using a digital coherent receiver. *Optics Express*, Optical Society of America, v. 18, n. 25, p. 25855–25860, 2010. Citado na página 21.

MECOZZI, A.; ESSIAMBRE, R.-J. Nonlinear shannon limit in pseudolinear coherent systems. *Journal of Lightwave Technology*, IEEE, v. 30, n. 12, p. 2011–2024, 2012. Citado na página 18.

MEHRA, M.; SADAWARTI, H.; SINGH, M. Performance analysis of coherent optical communication system for higher order dual polarization modulation formats. *Optik*, Elsevier, v. 135, p. 174–179, 2017. Citado na página 18.

MITRA, P. P.; STARK, J. B. Nonlinear limits to the information capacity of optical fibre communications. *Nature*, Nature Publishing Group, v. 411, n. 6841, p. 1027, 2001. Citado na página 18.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015. Citado 5 vezes nas páginas 43, 44, 45, 49 e 50.

ROTTONDI, C.; BARLETTA, L.; GIUSTI, A.; TORNATORE, M. Machine-learning method for quality of transmission prediction of unestablished lightpaths. *Journal of Optical Communications and Networking*, Optical Society of America, v. 10, n. 2, p. A286–A297, 2018. Citado na página 19.

SAMADI, P.; AMAR, D.; LEPERS, C.; LOURDIANE, M.; BERGMAN, K. Quality of transmission prediction with machine learning for dynamic operation of optical WDM networks. In: IEEE. *2017 European Conference on Optical Communication (ECOC)*. [S.l.], 2017. p. 1–3. Citado na página 19.

SANFERRARE, R. J. Terrestrial lightwave systems. *AT&T Technical Journal*, Nokia Bell Labs, v. 66, n. 1, p. 95–107, 1987. Citado na página 17.

SCHMOGROW, R.; NEBENDAHL, B.; WINTER, M.; JOSTEN, A.; HILLERKUSS, D.; KOENIG, S.; MEYER, J.; DRESCHMANN, M.; HUEBNER, M.; KOOS, C. *et al.* Error vector magnitude as a performance measure for advanced modulation formats. *IEEE Photonics Technology Letters*, IEEE, v. 24, n. 1, p. 61–63, 2011. Citado 2 vezes nas páginas 29 e 30.

SHAFIK, R. A.; RAHMAN, M. S.; ISLAM, A. R. On the extended relationships among EVM, BER and SNR as performance metrics. In: IEEE. *2006 International Conference on Electrical and Computer Engineering*. [S.l.], 2006. p. 408–411. Citado na página 29.

SHANNON, C. E. A mathematical theory of communication. *Bell System Technical Journal*, Wiley Online Library, v. 27, n. 3, p. 379–423, 1948. Citado na página 18.

SHARMA, S.; SHARMA, S. Activation functions in neural networks. *Towards Data Science*, v. 6, n. 12, p. 310–316, 2017. Citado 2 vezes nas páginas 32 e 34.

SILVA, I. d.; SPATTI, D. H. Redes neurais artificiais para engenharia e ciências aplicadas. *São Paulo: Artliber*, v. 23, n. 5, p. 33–111, 2010. Citado 4 vezes nas páginas 36, 39, 41 e 42.

SILVA, L. M. da; PAULA, R. de; OLIVEIRA, J. A. de; SANTOS, M.; PENCHEL, R.; PEREZ, G. G.; ABBADE, M. L.; ALDAYA, I. Nonlinear phase noise compensation in single-span digital coherent optical systems employing artificial neural networks. In: IEEE. *2021 SBFoton International Optics and Photonics Conference (SBFoton IOPC)*. [S.l.], 2021. p. 1–4. Citado na página 19.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 59.

STOKES, M. "A standard default colour space for the internet-srgb", version 1.10. *http://www. w3. org/pub/WWW/Graphics/Color/sRGB. html*, 1996. Citado na página 44.

TORRES, J. J. G.; CHIUCHIARELLI, A.; THOMAS, V. A.; RALPH, S. E.; SOTO, A. M. C.; GONZÁLEZ, N. G. Adaptive nonsymmetrical demodulation based on machine learning to mitigate time-varying impairments. In: IEEE. *2016 IEEE Avionics and Vehicle Fiber-Optics and Photonics Conference (AVFOP)*. [S.l.], 2016. p. 289–290. Citado 2 vezes nas páginas 19 e 24.

WIDROW, B.; HOFF, M. E. *Adaptive Switching Circuits*. [S.l.], 1960. Citado 2 vezes nas páginas 39 e 40.

WU, J. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, v. 5, n. 23, p. 495, 2017. Citado 3 vezes nas páginas 46, 49 e 50.

WU, X.; JARGON, J. A.; SKOOG, R. A.; PARASCHIS, L.; WILLNER, A. E. Applications of artificial neural networks in optical performance monitoring. *Journal of Lightwave Technology*, IEEE, v. 27, n. 16, p. 3580–3589, 2009. Citado na página 19.

XU, B.; WANG, N.; CHEN, T.; LI, M. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. Citado 3 vezes nas páginas 34, 35 e 45.

YAMASHITA, R.; NISHIO, M.; DO, R. K. G.; TOGASHI, K. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, Springer, v. 9, n. 4, p. 611–629, 2018. Citado 4 vezes nas páginas 43, 44, 46 e 49.

Anexo A – Extrutura CNNs

Tabela 13 – Estrutura original de uma CNN MobileNetV2

Nome	Dim. saída	Param #
input_16 (InputLayer)	[(None, 224, 224, 3)]	0
Conv1_pad (ZeroPadding2D)	(None, 225, 225, 3)	0
Conv1 (Conv2D)	(None, 112, 112, 32)	864
bn_Conv1 (BatchNormalization)	(None, 112, 112, 32)	128
Conv1_relu (ReLU)	(None, 112, 112, 32)	0
expanded_conv_depthwise (DepthwiseConv2D)	(None, 112, 112, 32)	288
expanded_conv_depthwise_BN (BatchNormalization)	(None, 112, 112, 32)	128
expanded_conv_depthwise_relu (ReLU)	(None, 112, 112, 32)	0
expanded_conv_project (Conv2D)	(None, 112, 112, 16)	512
expanded_conv_project_BN (BatchNormalization)	(None, 112, 112, 16)	64
block_1_expand (Conv2D)	(None, 112, 112, 96)	1536
block_1_expand_BN (BatchNormalization)	(None, 112, 112, 96)	384
block_1_expand_relu (ReLU)	(None, 112, 112, 96)	0
block_1_pad (ZeroPadding2D)	(None, 113, 113, 96)	0
block_1_depthwise (DepthwiseConv2D)	(None, 56, 56, 96)	864
block_1_depthwise_BN (BatchNormalization)	(None, 56, 56, 96)	384
block_1_depthwise_relu (ReLU)	(None, 56, 56, 96)	0
block_1_project (Conv2D)	(None, 56, 56, 24)	2304

block_1_project_BN (BatchNormalization)	(None, 56, 56, 24)	96
block_2_expand (Conv2D)	(None, 56, 56, 144)	3456
block_2_expand_BN (BatchNormalization)	(None, 56, 56, 144)	576
block_2_expand_relu (ReLU)	(None, 56, 56, 144)	0
block_2_depthwise (DepthwiseConv2D)	(None, 56, 56, 144)	1296
block_2_depthwise_BN (BatchNormalization)	(None, 56, 56, 144)	576
block_2_depthwise_relu (ReLU)	(None, 56, 56, 144)	0
block_2_project (Conv2D)	(None, 56, 56, 24)	3456
block_2_project_BN (BatchNormalization)	(None, 56, 56, 24)	96
block_2_add (Add)	(None, 56, 56, 24)	0
block_3_expand (Conv2D)	(None, 56, 56, 144)	3456
block_3_expand_BN (BatchNormalization)	(None, 56, 56, 144)	576
block_3_expand_relu (ReLU)	(None, 56, 56, 144)	0
block_3_pad (ZeroPadding2D)	(None, 57, 57, 144)	0
block_3_depthwise (DepthwiseConv2D)	(None, 28, 28, 144)	1296
block_3_depthwise_BN (BatchNormalization)	(None, 28, 28, 144)	576
block_3_depthwise_relu (ReLU)	(None, 28, 28, 144)	0
block_3_project (Conv2D)	(None, 28, 28, 32)	4608
block_3_project_BN (BatchNormalization)	(None, 28, 28, 32)	128
block_4_expand (Conv2D)	(None, 28, 28, 192)	6144
block_4_expand_BN (BatchNormalization)	(None, 28, 28, 192)	768

block_4_expand_relu (ReLU)	(None, 28, 28, 192)	0
block_4_depthwise (DepthwiseConv2D)	(None, 28, 28, 192)	1728
block_4_depthwise_BN (BatchNormalization)	(None, 28, 28, 192)	768
block_4_depthwise_relu (ReLU)	(None, 28, 28, 192)	0
block_4_project (Conv2D)	(None, 28, 28, 32)	6144
block_4_project_BN (BatchNormalization)	(None, 28, 28, 32)	128
block_4_add (Add)	(None, 28, 28, 32)	0
block_5_expand (Conv2D)	(None, 28, 28, 192)	6144
block_5_expand_BN (BatchNormalization)	(None, 28, 28, 192)	768
block_5_expand_relu (ReLU)	(None, 28, 28, 192)	0
block_5_depthwise (DepthwiseConv2D)	(None, 28, 28, 192)	1728
block_5_depthwise_BN (BatchNormalization)	(None, 28, 28, 192)	768
block_5_depthwise_relu (ReLU)	(None, 28, 28, 192)	0
block_5_project (Conv2D)	(None, 28, 28, 32)	6144
block_5_project_BN (BatchNormalization)	(None, 28, 28, 32)	128
block_5_add (Add)	(None, 28, 28, 32)	0
block_6_expand (Conv2D)	(None, 28, 28, 192)	6144
block_6_expand_BN (BatchNormalization)	(None, 28, 28, 192)	768
block_6_expand_relu (ReLU)	(None, 28, 28, 192)	0
block_6_pad (ZeroPadding2D)	(None, 29, 29, 192)	0
block_6_depthwise (DepthwiseConv2D)	(None, 14, 14, 192)	1728

block_6_depthwise_BN (BatchNormalization)	(None, 14, 14, 192)	768
block_6_depthwise_relu (ReLU)	(None, 14, 14, 192)	0
block_6_project (Conv2D)	(None, 14, 14, 64)	12288
block_6_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_7_expand (Conv2D)	(None, 14, 14, 384)	24576
block_7_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_7_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_7_depthwise (DepthwiseConv2D)	(None, 14, 14, 384)	3456
block_7_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_7_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_7_project (Conv2D)	(None, 14, 14, 64)	24576
block_7_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_7_add (Add)	(None, 14, 14, 64)	0
block_8_expand (Conv2D)	(None, 14, 14, 384)	24576
block_8_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_8_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_8_depthwise (DepthwiseConv2D)	(None, 14, 14, 384)	3456
block_8_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_8_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_8_project (Conv2D)	(None, 14, 14, 64)	24576
block_8_project_BN (BatchNormalization)	(None, 14, 14, 64)	256

block_8_add (Add)	(None, 14, 14, 64)	0
block_9_expand (Conv2D)	(None, 14, 14, 384)	24576
block_9_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_9_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_9_depthwise (DepthwiseConv2D)	(None, 14, 14, 384)	3456
block_9_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_9_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_9_project (Conv2D)	(None, 14, 14, 64)	24576
block_9_project_BN (BatchNormalization)	(None, 14, 14, 64)	256
block_9_add (Add)	(None, 14, 14, 64)	0
block_10_expand (Conv2D)	(None, 14, 14, 384)	24576
block_10_expand_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_10_expand_relu (ReLU)	(None, 14, 14, 384)	0
block_10_depthwise (DepthwiseConv2D)	(None, 14, 14, 384)	3456
block_10_depthwise_BN (BatchNormalization)	(None, 14, 14, 384)	1536
block_10_depthwise_relu (ReLU)	(None, 14, 14, 384)	0
block_10_project (Conv2D)	(None, 14, 14, 96)	36864
block_10_project_BN (BatchNormalization)	(None, 14, 14, 96)	384
block_11_expand (Conv2D)	(None, 14, 14, 576)	55296
block_11_expand_BN (BatchNormalization)	(None, 14, 14, 576)	2304
block_11_expand_relu (ReLU)	(None, 14, 14, 576)	0

block_11_depthwise (DepthwiseConv2D)	(None, 14, 14, 576)	5184
block_11_depthwise_BN (BatchNormalization)	(None, 14, 14, 576)	2304
block_11_depthwise_relu (ReLU)	(None, 14, 14, 576)	0
block_11_project (Conv2D)	(None, 14, 14, 96)	55296
block_11_project_BN (BatchNormalization)	(None, 14, 14, 96)	384
block_11_add (Add)	(None, 14, 14, 96)	0
block_12_expand (Conv2D)	(None, 14, 14, 576)	55296
block_12_expand_BN (BatchNormalization)	(None, 14, 14, 576)	2304
block_12_expand_relu (ReLU)	(None, 14, 14, 576)	0
block_12_depthwise (DepthwiseConv2D)	(None, 14, 14, 576)	5184
block_12_depthwise_BN (BatchNormalization)	(None, 14, 14, 576)	2304
block_12_depthwise_relu (ReLU)	(None, 14, 14, 576)	0
block_12_project (Conv2D)	(None, 14, 14, 96)	55296
block_12_project_BN (BatchNormalization)	(None, 14, 14, 96)	384
block_12_add (Add)	(None, 14, 14, 96)	0
block_13_expand (Conv2D)	(None, 14, 14, 576)	55296
block_13_expand_BN (BatchNormalization)	(None, 14, 14, 576)	2304
block_13_expand_relu (ReLU)	(None, 14, 14, 576)	0
block_13_pad (ZeroPadding2D)	(None, 15, 15, 576)	0
block_13_depthwise (DepthwiseConv2D)	(None, 7, 7, 576)	5184
block_13_depthwise_BN (BatchNormalization)	(None, 7, 7, 576)	2304

block_13_depthwise_relu (ReLU)	(None, 7, 7, 576)	0
block_13_project (Conv2D)	(None, 7, 7, 160)	92160
block_13_project_BN (BatchNormalization)	(None, 7, 7, 160)	640
block_14_expand (Conv2D)	(None, 7, 7, 960)	153600
block_14_expand_BN (BatchNormalization)	(None, 7, 7, 960)	3840
block_14_expand_relu (ReLU)	(None, 7, 7, 960)	0
block_14_depthwise (DepthwiseConv2D)	(None, 7, 7, 960)	8640
block_14_depthwise_BN (BatchNormalization)	(None, 7, 7, 960)	3840
block_14_depthwise_relu (ReLU)	(None, 7, 7, 960)	0
block_14_project (Conv2D)	(None, 7, 7, 160)	153600
block_14_project_BN (BatchNormalization)	(None, 7, 7, 160)	640
block_14_add (Add)	(None, 7, 7, 160)	0
block_15_expand (Conv2D)	(None, 7, 7, 960)	153600
block_15_expand_BN (BatchNormalization)	(None, 7, 7, 960)	3840
block_15_expand_relu (ReLU)	(None, 7, 7, 960)	0
block_15_depthwise (DepthwiseConv2D)	(None, 7, 7, 960)	8640
block_15_depthwise_BN (BatchNormalization)	(None, 7, 7, 960)	3840
block_15_depthwise_relu (ReLU)	(None, 7, 7, 960)	0
block_15_project (Conv2D)	(None, 7, 7, 160)	153600
block_15_project_BN (BatchNormalization)	(None, 7, 7, 160)	640
block_15_add (Add)	(None, 7, 7, 160)	0

block_16_expand (Conv2D)	(None, 7, 7, 960)	153600
block_16_expand_BN (BatchNormalization)	(None, 7, 7, 960)	3840
block_16_expand_relu (ReLU)	(None, 7, 7, 960)	0
block_16_depthwise (DepthwiseConv2D)	(None, 7, 7, 960)	8640
block_16_depthwise_BN (BatchNormalization)	(None, 7, 7, 960)	3840
block_16_depthwise_relu (ReLU)	(None, 7, 7, 960)	0
block_16_project (Conv2D)	(None, 7, 7, 320)	307200
block_16_project_BN (BatchNormalization)	(None, 7, 7, 320)	1280
Conv_1 (Conv2D)	(None, 7, 7, 1280)	409600
Conv_1_bn (BatchNormalization)	(None, 7, 7, 1280)	5120
out_relu (ReLU)	(None, 7, 7, 1280)	0
global_average_pooling2d_13 (GlobalAveragePooling2D)	(None, 1280)	0
Logits (Dense)	(None, 1000)	1281000

Total de parâmetros: 3.538.984

Parâmetros treináveis: 3.504.872

Parâmetros não treináveis: 34.112

Tabela 14 – Estrutura da CNN MobileNetV2 utilizada para estimar valores de BER

Tipo de camada	Dim. saída	Param #
input_15 (InputLayer)	[(None, 128, 128, 3)]	0
Conv1_pad (ZeroPadding2D)	(None, 129, 129, 3)	0
Conv1 (Conv2D)	(None, 64, 64, 32)	864
bn_Conv1 (BatchNormalization)	(None, 64, 64, 32)	128
Conv1_relu (ReLU)	(None, 64, 64, 32)	0
expanded_conv_depthwise (DepthwiseConv2D)	(None, 64, 64, 32)	288
expanded_conv_depthwise_BN (BatchNormalization)	(None, 64, 64, 32)	128
expanded_conv_depthwise_relu (ReLU)	(None, 64, 64, 32)	0
expanded_conv_project (Conv2D)	(None, 64, 64, 16)	512
expanded_conv_project_BN (BatchNormalization)	(None, 64, 64, 16)	64
block_1_expand (Conv2D)	(None, 64, 64, 96)	1536
block_1_expand_BN (BatchNormalization)	(None, 64, 64, 96)	384
block_1_expand_relu (ReLU)	(None, 64, 64, 96)	0
block_1_pad (ZeroPadding2D)	(None, 65, 65, 96)	0
block_1_depthwise (DepthwiseConv2D)	(None, 32, 32, 96)	864
block_1_depthwise_BN (BatchNormalization)	(None, 32, 32, 96)	384
block_1_depthwise_relu (ReLU)	(None, 32, 32, 96)	0
block_1_project (Conv2D)	(None, 32, 32, 24)	2304
block_1_project_BN (BatchNormalization)	(None, 32, 32, 24)	96

block_2_expand (Conv2D)	(None, 32, 32, 144)	3456
block_2_expand_BN (BatchNormalization)	(None, 32, 32, 144)	576
block_2_expand_relu (ReLU)	(None, 32, 32, 144)	0
block_2_depthwise (DepthwiseConv2D)	(None, 32, 32, 144)	1296
block_2_depthwise_BN (BatchNormalization)	(None, 32, 32, 144)	576
block_2_depthwise_relu (ReLU)	(None, 32, 32, 144)	0
block_2_project (Conv2D)	(None, 32, 32, 24)	3456
block_2_project_BN (BatchNormalization)	(None, 32, 32, 24)	96
block_2_add (Add)	(None, 32, 32, 24)	0
block_3_expand (Conv2D)	(None, 32, 32, 144)	3456
block_3_expand_BN (BatchNormalization)	(None, 32, 32, 144)	576
block_3_expand_relu (ReLU)	(None, 32, 32, 144)	0
block_3_pad (ZeroPadding2D)	(None, 33, 33, 144)	0
block_3_depthwise (DepthwiseConv2D)	(None, 16, 16, 144)	1296
block_3_depthwise_BN (BatchNormalization)	(None, 16, 16, 144)	576
block_3_depthwise_relu (ReLU)	(None, 16, 16, 144)	0
block_3_project (Conv2D)	(None, 16, 16, 32)	4608
block_3_project_BN (BatchNormalization)	(None, 16, 16, 32)	128
block_4_expand (Conv2D)	(None, 16, 16, 192)	6144
block_4_expand_BN (BatchNormalization)	(None, 16, 16, 192)	768
block_4_expand_relu (ReLU)	(None, 16, 16, 192)	0

block_4_depthwise (DepthwiseConv2D)	(None, 16, 16, 192)	1728
block_4_depthwise_BN (BatchNormalization)	(None, 16, 16, 192)	768
block_4_depthwise_relu (ReLU)	(None, 16, 16, 192)	0
block_4_project (Conv2D)	(None, 16, 16, 32)	6144
block_4_project_BN (BatchNormalization)	(None, 16, 16, 32)	128
block_4_add (Add)	(None, 16, 16, 32)	0
block_5_expand (Conv2D)	(None, 16, 16, 192)	6144
block_5_expand_BN (BatchNormalization)	(None, 16, 16, 192)	768
block_5_expand_relu (ReLU)	(None, 16, 16, 192)	0
block_5_depthwise (DepthwiseConv2D)	(None, 16, 16, 192)	1728
block_5_depthwise_BN (BatchNormalization)	(None, 16, 16, 192)	768
block_5_depthwise_relu (ReLU)	(None, 16, 16, 192)	0
block_5_project (Conv2D)	(None, 16, 16, 32)	6144
block_5_project_BN (BatchNormalization)	(None, 16, 16, 32)	128
block_5_add (Add)	(None, 16, 16, 32)	0
block_6_expand (Conv2D)	(None, 16, 16, 192)	6144
block_6_expand_BN (BatchNormalization)	(None, 16, 16, 192)	768
block_6_expand_relu (ReLU)	(None, 16, 16, 192)	0
block_6_pad (ZeroPadding2D)	(None, 17, 17, 192)	0
block_6_depthwise (DepthwiseConv2D)	(None, 8, 8, 192)	1728
block_6_depthwise_BN (BatchNormalization)	(None, 8, 8, 192)	768

block_6_depthwise_relu (ReLU)	(None, 8, 8, 192)	0
block_6_project (Conv2D)	(None, 8, 8, 64)	12288
block_6_project_BN (BatchNormalization)	(None, 8, 8, 64)	256
block_7_expand (Conv2D)	(None, 8, 8, 384)	24576
block_7_expand_BN (BatchNormalization)	(None, 8, 8, 384)	1536
block_7_expand_relu (ReLU)	(None, 8, 8, 384)	0
block_7_depthwise (DepthwiseConv2D)	(None, 8, 8, 384)	3456
block_7_depthwise_BN (BatchNormalization)	(None, 8, 8, 384)	1536
block_7_depthwise_relu (ReLU)	(None, 8, 8, 384)	0
block_7_project (Conv2D)	(None, 8, 8, 64)	24576
block_7_project_BN (BatchNormalization)	(None, 8, 8, 64)	256
block_7_add (Add)	(None, 8, 8, 64)	0
block_8_expand (Conv2D)	(None, 8, 8, 384)	24576
block_8_expand_BN (BatchNormalization)	(None, 8, 8, 384)	1536
block_8_expand_relu (ReLU)	(None, 8, 8, 384)	0
block_8_depthwise (DepthwiseConv2D)	(None, 8, 8, 384)	3456
block_8_depthwise_BN (BatchNormalization)	(None, 8, 8, 384)	1536
block_8_depthwise_relu (ReLU)	(None, 8, 8, 384)	0
block_8_project (Conv2D)	(None, 8, 8, 64)	24576
block_8_project_BN (BatchNormalization)	(None, 8, 8, 64)	256
block_8_add (Add)	(None, 8, 8, 64)	0

block_9_expand (Conv2D)	(None, 8, 8, 384)	24576
block_9_expand_BN (BatchNormalization)	(None, 8, 8, 384)	1536
block_9_expand_relu (ReLU)	(None, 8, 8, 384)	0
block_9_depthwise (DepthwiseConv2D)	(None, 8, 8, 384)	3456
block_9_depthwise_BN (BatchNormalization)	(None, 8, 8, 384)	1536
block_9_depthwise_relu (ReLU)	(None, 8, 8, 384)	0
block_9_project (Conv2D)	(None, 8, 8, 64)	24576
block_9_project_BN (BatchNormalization)	(None, 8, 8, 64)	256
block_9_add (Add)	(None, 8, 8, 64)	0
block_10_expand (Conv2D)	(None, 8, 8, 384)	24576
block_10_expand_BN (BatchNormalization)	(None, 8, 8, 384)	1536
block_10_expand_relu (ReLU)	(None, 8, 8, 384)	0
block_10_depthwise (DepthwiseConv2D)	(None, 8, 8, 384)	3456
block_10_depthwise_BN (BatchNormalization)	(None, 8, 8, 384)	1536
block_10_depthwise_relu (ReLU)	(None, 8, 8, 384)	0
block_10_project (Conv2D)	(None, 8, 8, 96)	36864
block_10_project_BN (BatchNormalization)	(None, 8, 8, 96)	384
block_11_expand (Conv2D)	(None, 8, 8, 576)	55296
block_11_expand_BN (BatchNormalization)	(None, 8, 8, 576)	2304
block_11_expand_relu (ReLU)	(None, 8, 8, 576)	0
block_11_depthwise (DepthwiseConv2D)	(None, 8, 8, 576)	5184

block_11_depthwise_BN (BatchNormalization)	(None, 8, 8, 576)	2304
block_11_depthwise_relu (ReLU)	(None, 8, 8, 576)	0
block_11_project (Conv2D)	(None, 8, 8, 96)	55296
block_11_project_BN (BatchNormalization)	(None, 8, 8, 96)	384
block_11_add (Add)	(None, 8, 8, 96)	0
block_12_expand (Conv2D)	(None, 8, 8, 576)	55296
block_12_expand_BN (BatchNormalization)	(None, 8, 8, 576)	2304
block_12_expand_relu (ReLU)	(None, 8, 8, 576)	0
block_12_depthwise (DepthwiseConv2D)	(None, 8, 8, 576)	5184
block_12_depthwise_BN (BatchNormalization)	(None, 8, 8, 576)	2304
block_12_depthwise_relu (ReLU)	(None, 8, 8, 576)	0
block_12_project (Conv2D)	(None, 8, 8, 96)	55296
block_12_project_BN (BatchNormalization)	(None, 8, 8, 96)	384
block_12_add (Add)	(None, 8, 8, 96)	0
block_13_expand (Conv2D)	(None, 8, 8, 576)	55296
block_13_expand_BN (BatchNormalization)	(None, 8, 8, 576)	2304
block_13_expand_relu (ReLU)	(None, 8, 8, 576)	0
block_13_pad (ZeroPadding2D)	(None, 9, 9, 576)	0
block_13_depthwise (DepthwiseConv2D)	(None, 4, 4, 576)	5184
block_13_depthwise_BN (BatchNormalization)	(None, 4, 4, 576)	2304
block_13_depthwise_relu (ReLU)	(None, 4, 4, 576)	0

block_13_project (Conv2D)	(None, 4, 4, 160)	92160
block_13_project_BN (BatchNormalization)	(None, 4, 4, 160)	640
block_14_expand (Conv2D)	(None, 4, 4, 960)	153600
block_14_expand_BN (BatchNormalization)	(None, 4, 4, 960)	3840
block_14_expand_relu (ReLU)	(None, 4, 4, 960)	0
block_14_depthwise (DepthwiseConv2D)	(None, 4, 4, 960)	8640
block_14_depthwise_BN (BatchNormalization)	(None, 4, 4, 960)	3840
block_14_depthwise_relu (ReLU)	(None, 4, 4, 960)	0
block_14_project (Conv2D)	(None, 4, 4, 160)	153600
block_14_project_BN (BatchNormalization)	(None, 4, 4, 160)	640
block_14_add (Add)	(None, 4, 4, 160)	0
block_15_expand (Conv2D)	(None, 4, 4, 960)	153600
block_15_expand_BN (BatchNormalization)	(None, 4, 4, 960)	3840
block_15_expand_relu (ReLU)	(None, 4, 4, 960)	0
block_15_depthwise (DepthwiseConv2D)	(None, 4, 4, 960)	8640
block_15_depthwise_BN (BatchNormalization)	(None, 4, 4, 960)	3840
block_15_depthwise_relu (ReLU)	(None, 4, 4, 960)	0
block_15_project (Conv2D)	(None, 4, 4, 160)	153600
block_15_project_BN (BatchNormalization)	(None, 4, 4, 160)	640
block_15_add (Add)	(None, 4, 4, 160)	0
block_16_expand (Conv2D)	(None, 4, 4, 960)	153600

block_16_expand_BN (BatchNormalization)	(None, 4, 4, 960)	3840
block_16_expand_relu (ReLU)	(None, 4, 4, 960)	0
block_16_depthwise (DepthwiseConv2D)	(None, 4, 4, 960)	8640
block_16_depthwise_BN (BatchNormalization)	(None, 4, 4, 960)	3840
block_16_depthwise_relu (ReLU)	(None, 4, 4, 960)	0
block_16_project (Conv2D)	(None, 4, 4, 320)	307200
block_16_project_BN (BatchNormalization)	(None, 4, 4, 320)	1280
Conv_1 (Conv2D)	(None, 4, 4, 1280)	409600
Conv_1_bn (BatchNormalization)	(None, 4, 4, 1280)	5120
out_relu (ReLU)	(None, 4, 4, 1280)	0
global_average_pooling2d_12 (GlobalAveragePooling2D)	(None, 1280)	0
Logits (Dense)	(None, 1)	1281

Total de parâmetros: 2.259.265

Parâmetros treináveis: 2.225.153

Parâmetros não treináveis: 34.112