

Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Pedro Vinícius Almeida Borges de Venâncio

**Um sistema automático de detecção de incêndios
baseado em aprendizado profundo para dispositivos
de baixo poder computacional**

Belo Horizonte

Julho de 2021

Pedro Vinícius Almeida Borges de Venâncio

**Um sistema automático de detecção de incêndios
baseado em aprendizado profundo para dispositivos
de baixo poder computacional**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Adriano Vilela Barbosa
Coorientador: Adriano Chaves Lisboa

Belo Horizonte

Julho de 2021

V448s

Venâncio, Pedro Vinícius Almeida Borges de.

Um sistema automático de detecção de incêndios baseado em aprendizado profundo para dispositivos de baixo poder computacional [recurso eletrônico] / Pedro Vinícius Almeida Borges de Venâncio. - 2021.
1 recurso online (153 f. : il., color.) : pdf.

Orientador: Adriano Vilela Barbosa.

Coorientador: Adriano Chaves Lisboa.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 137-153.

Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Aprendizado profundo - Teses.
3. Incêndios - Teses. 4. Otimização multiobjetivo - Teses. 5. Redes neurais convolucionais - Teses. 6. Visão por computador - Teses.
I. Barbosa, Adriano Vilela. II. Lisboa, Adriano Chaves. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

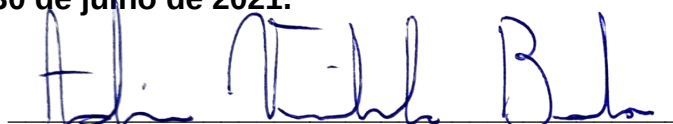
"Um Sistema Automático de Detecção de Incêndios Baseado em Aprendizado Profundo Para Dispositivos de Baixo Poder Computacional"

Pedro Vinícius Almeida Borges de Venâncio


Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 30 de julho de 2021.

Por:



Prof. Dr. Adriano Vilela Barbosa
DELT (UFMG) - Orientador



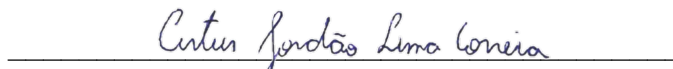
Dr. Adriano Chaves Lisboa
Gaia Soluções em Engenharia - Co-orientador



Prof. Dr. Hani Camille Yehia
DELT (UFMG)



Prof. Dr. Cristiano Leite de Castro
DEE (UFMG)



Dr. Artur Jordão Lima Correia
CEPETRO (Unicamp)

Dedico este trabalho a todos os docentes do Brasil, que mesmo diante de uma contínua desvalorização salarial e de péssimas condições de trabalho perseveraram na nobre missão de educar, ensinar e formar cidadãos mais humanos.

Agradecimentos

Para começar, gostaria de relatar minha imensa gratidão a Deus por me amparar desde o princípio dos meus estudos, quando eu ainda sequer imaginava que teria o privilégio de defender um título de mestre numa das universidades federais mais conceituadas do Brasil. Confesso que durante o percurso fraquejei inúmeras vezes e questioneei os seus propósitos, mas sei que o Senhor sempre esteve comigo.

“Esforça-te, e tem bom ânimo; não temas, nem te espantes; porque o Senhor teu Deus é contigo, por onde quer que andares.” Josué 1:9.

Agradeço aos meus pais José Silvano Borges e Mércia das Graças Almeida Borges por todo amor dedicado a mim e por todo suporte emocional e financeiro. Vocês me ensinaram a ter disciplina desde cedo e a perseverar independentemente das circunstâncias, virtudes as quais julgo essenciais para conquistar os meus objetivos. Meus sinceros agradecimentos também ao meu irmão Davi Almeida Borges de Venâncio, cujas necessidades especiais me motivam diariamente a buscar minha melhor versão, como ser humano e como profissional. Darei cem por cento de mim todos os dias da minha vida para lhe garantir um futuro digno e sereno, bem como para ajudar outros portadores da Síndrome de Prader-Willi.

À minha companheira de vida Thais Macela de Lira Menegaldi, meus singelos agradecimentos por todo apoio e incentivo durante esse período de estresse contínuo, repleto de lamentações e frustrações. Sem sua ternura e prudência eu jamais teria discernimento suficiente para superar os diversos obstáculos que surgiram na condução da minha pesquisa. Sou grato também às minhas avós, meus tios, meus primos e meus amigos por todo carinho e compreensão no decorrer desse processo.

Finalmente, agradeço ao meu orientador Adriano Vilela Barbosa por todas as sugestões, sempre muito pertinentes, e por todas as assistências. Ao meu coorientador Adriano Chaves Lisboa por todos os ensinamentos e por acreditar que poderíamos fazer juntos um trabalho relevante em um tema inicialmente desconhecido por nós dois. Ao meu ex-colega de empresa Lucas Gonçalves Bispo pela grande ajuda na elaboração da base de dados e por toda paciência. À Gaia, à CEMIG e ao CEFALA pela infraestrutura e pelo suporte financeiro, sem os quais esse trabalho não seria possível.

“Em questões de ciência, a autoridade de milhares não vale o humilde raciocínio de um único indivíduo.”
(Galileu Galilei)

Resumo

Incêndios de grandes proporções vêm sendo frequentemente noticiados pelas imprensas nacional e internacional nos últimos anos. Os impactos decorrentes de tais catástrofes compreendem uma série de consequências irreversíveis, como perda da biodiversidade, maior emissão de gases poluentes na atmosfera, efeitos deletérios para a saúde humana e destruição de propriedades rurais e patrimônios culturais. Sob essa perspectiva, torna-se indispensável a busca por soluções eficazes de prevenção e combate a incêndios. Uma potencial solução para esse dilema seria um sistema autônomo baseado em visão computacional capaz de identificar focos de incêndio rapidamente, viabilizando a supressão para atenuação dos danos e, conseqüentemente, minimizando os custos operacionais de combate e restauração. O estado da arte destes sistemas utiliza redes neurais convolucionais para reconhecimento dos principais indicadores visuais de incêndio: fogo e fumaça. Entretanto, algoritmos de aprendizado profundo como esse são computacionalmente caros, apresentam milhares de parâmetros, consomem uma quantidade considerável de memória e ainda requerem um grande volume de dados rotulados para treinamento. Perante o exposto, neste trabalho é apresentada uma ferramenta de detecção automática de incêndios para dispositivos móveis e com processamento limitado. O modelo de detecção de fogo e fumaça é desenvolvido a partir do treinamento de uma rede neural convolucional em uma base de dados inédita, a qual compreende uma variedade de eventos reais de incêndio. Posteriormente, filtros convolucionais menos relevantes desse modelo são identificados e removidos, de modo a preservar o desempenho de detecção obtido em uma arquitetura otimizada. Os resultados experimentais mostram que é possível construir um detector de incêndios baseado em aprendizado profundo que seja não apenas robusto mas também computacionalmente eficiente, além de instigar um cenário de monitoramento ambiental em larga escala com processamento local distribuído e de baixo custo de infraestrutura.

Palavras-chave: Aprendizado profundo, detecção de incêndios em larga escala, otimização multiobjetivo, redes neurais convolucionais e visão computacional.

Abstract

Large-scale fires have been frequently reported both in the national and the international press in recent years. The impacts resulting from such catastrophes comprise a series of irreversible consequences, such as biodiversity losses, greater emission of polluting gases into the atmosphere, deleterious effects on human health and destruction of rural properties and cultural heritage. From this perspective, it is essential to search for effective solutions for preventing and fighting fires. A potential solution to this dilemma is an autonomous computer vision system capable of quickly identifying fire outbreaks, enabling suppression to mitigate damages and, consequently, minimizing combat and restoration operating costs. The state of the art of these systems use convolutional neural networks to recognize the main visual indicators of wildfires: fire and smoke. However, deep learning algorithms such as these are computationally expensive, have thousands of parameters, consume a considerable amount of memory and require a large volume of labeled data for training. In this context, this work presents an automatic fire detection tool aimed at low end, mobile computing devices. The fire and smoke detection model is derived from the training of a convolutional neural network on a novel database, which comprises a variety of real fire events. Subsequently, less relevant convolutional filters in the model are identified and removed, in order to preserve the detection performance obtained in an optimized architecture. The experimental results show that it is possible to build a fire detector based on deep learning that is both robust and computationally efficient, in addition to instigating a large-scale environmental monitoring scenario with distributed local processing and low infrastructure cost.

Keywords: Deep learning, large-scale fire detection, multi-objective optimization, convolutional neural networks and computer vision.

Lista de figuras

1.1	Relação de compromisso entre o custo computacional e o número de filtros convolucionais de algumas CNNs projetadas para tarefas de classificação e detecção.	20
3.1	Exemplo de validação cruzada <i>k-fold</i> com $k = 5$	41
3.2	Relação entre a capacidade do modelo e os erros de treinamento e teste. . .	42
3.3	Modelos com diferentes capacidades em um problema de regressão polinomial. .	43
3.4	Modelos com diferentes capacidades em um problema de classificação binária. .	43
3.5	Pontos críticos de uma função de custo $f(s)$	45
3.6	Comparação entre as trajetórias do gradiente descendente e do gradiente descendente em lote.	47
3.7	Impacto do <i>momentum</i> na trajetória do gradiente descendente em lote. . .	48
3.8	Tipos de <i>learning rate schedulers</i>	50
3.9	Impacto da técnica <i>warm-up</i> na taxa de aprendizado.	50
3.10	Comparação entre as topologias de uma rede de alimentação direta e de uma rede neural recorrente.	51
3.11	Representação clássica de uma rede neural com uma única camada intermediária.	52
3.12	Representação simplificada de uma rede neural profunda.	53
3.13	Função de ativação identidade (à esquerda) e sua derivada (à direita). . . .	56
3.14	Função de ativação sigmoide (à esquerda) e sua derivada (à direita). . . .	57
3.15	Função tangente hiperbólica (à esquerda) e sua derivada (à direita). . . .	59
3.16	Função ReLU (à esquerda) e sua derivada (à direita).	59
3.17	Função LReLU (à esquerda) e sua derivada (à direita).	60
3.18	Variações da ReLU (à esquerda) e suas respectivas derivadas (à direita). .	62
3.19	<i>Softplus</i> , <i>Swish</i> e <i>Mish</i> (à esquerda) e suas respectivas derivadas (à direita). .	64
3.20	Imagem como uma matriz de pixels.	70
3.21	Comparação entre uma rede neural totalmente conectada e uma rede neural esparsa.	71
3.22	Estrutura de um bloco residual.	75
3.23	Representação de um conjunto de filtros convolucionais como volume. . . .	76

3.24	Curva de precisão e revocação.	81
3.25	Estimativa da precisão média a partir da curva de precisão e revocação. . .	82
3.26	Grade de células $S \times S$ (à esquerda) e mapa de probabilidades de classe (à direita).	85
3.27	Supressão não-máxima e predição final.	85
3.28	Arquitetura simplificada do detector de objetos YOLOv3.	90
3.29	Representação simplificada da arquitetura do YOLOv4.	91
3.30	Módulos PAN e SAM.	92
4.1	Nuvem de palavras contendo a lista de vocábulos utilizados na busca por imagens de fogo e fumaça.	95
4.2	Incêndio em grandes áreas florestais (à esquerda) e queimada não controlada oriunda do manejo de pastagens para gado (à direita).	95
4.3	Incêndio resultante de um acidente de trânsito (à esquerda) e incêndio residencial provocado por falhas na instalação elétrica (à direita).	96
4.4	Incêndio registrado por avião comercial (à esquerda) e drone de combate a incêndios (à direita).	96
4.5	Cenários desafiadores. Reflexos solares incidindo sobre a câmera (à esquerda). Fumaça cônica pequena no lado inferior direito da imagem, que é ofuscada pela nuvem cujo formato é similar (à direita).	97
4.6	Registro de fumaça parcialmente camuflada pela vegetação (à esquerda) e registro de foco esparso integralmente camuflado pela vegetação (à direita).	98
4.7	Padrões de fumaça artificial.	98
4.8	Exemplo de imagens sintéticas produzidas com dois focos (à esquerda) e quatro focos (à direita).	98
4.9	Câmeras de vigilância do projeto P&D ANEEL D0619 da CEMIG.	99
4.10	Exemplo de rotulação de uma imagem com fogo e fumaça.	100
4.11	Poda de um filtro convolucional.	102
4.12	Processo iterativo de poda de filtros em redes neurais convolucionais. . . .	102
4.13	Representação dos mapas de recursos como vetores de características. . . .	104
5.1	Ajuste da taxa de aprendizado das redes YOLOv4 e Tiny YOLOv4 conforme o valor de mAP@0,50 no conjunto de validação.	114
5.2	Ajuste do limiar de confiança segundo os valores médios de F_1 e IoU nos 5 <i>folds</i> destinados à validação.	114
5.3	Detecções de fogo e fumaça em ambientes noturnos.	117
5.4	Detecções de focos de incêndio em estágios iniciais.	117

5.5	Detecções de fumaça com padrão cônico.	118
5.6	Detecções de múltiplos focos de fogo.	118
5.7	Detecção por vista aérea (à esquerda) e de focos sintéticos (à direita).	118
5.8	Aranha (à esquerda) e gotas de chuva (à direita) na lente da câmera.	118
5.9	Reflexos solares (à esquerda) e luzes artificiais (à direita).	119
5.10	Ambiente com névoa (à esquerda) e neblina com aspecto de fumaça (à direita).	119
5.11	Nuvens equivocadamente detectadas como fumaça.	119
5.12	Fumaças esparsas não detectadas.	119
5.13	Ajuste da taxa de aprendizado dos modelos podados com $p = 30\%$ e com $p = 80\%$ por cada uma das técnicas conforme o valor de mAP@0,50 no conjunto de validação.	123
5.14	Relações de compromisso obtidas pelas técnicas de poda consideradas.	124
5.15	Desempenho dos modelos podados com $p = 80\%$ por cada uma das técnicas de poda, segundo as métricas mAP@0,50 (à esquerda) e F_1 (à direita) no conjunto de teste, em função do número de iterações.	128
5.16	Relação do tempo de inicialização (à esquerda) e do tempo de detecção (à direita) no Raspberry Pi 4 com a intensidade de poda empregada.	130
5.17	Relação do desempenho dos modelos segundo a métrica mAP@0,50 no conjunto de teste com o tempo médio de inicialização (à esquerda) e com o tempo médio de detecção (à direita) no Raspberry Pi 4.	131

Lista de tabelas

3.1	Matriz de confusão com $C = 2$	79
3.2	Darknet-19 [Redmon e Farhadi, 2017].	89
3.3	Darknet-53 [Redmon e Farhadi, 2018].	89
4.1	Bases de dados para reconhecimento de fogo e fumaça.	94
4.2	Distribuição de imagens por categoria na base de dados D-Fire.	101
5.1	Desempenho das redes Tiny YOLOv4 e YOLOv4 no conjunto de teste. . .	115
5.2	Desempenho das redes segundo a medida F_1 no conjunto de teste ao des- considerar os múltiplos objetos por imagem.	116
5.3	Descrição das técnicas de poda de filtros convolucionais consideradas. . . .	121
5.4	Relação do custo computacional do modelo com a intensidade de poda em- pregada na arquitetura da rede.	122
5.5	Desempenho dos modelos podados pelas técnicas consideradas segundo o mAP@0,50 no conjunto de teste.	125
5.6	Desempenho dos modelos podados pelas técnicas consideradas segundo o AP@0,50 de fumaça no conjunto de teste.	125
5.7	Desempenho dos modelos podados pelas técnicas consideradas segundo o AP@0,50 de fogo no conjunto de teste.	125
5.8	Desempenho dos modelos podados pelas técnicas consideradas segundo a medida F_1 no conjunto de teste.	126
5.9	Desempenho dos modelos podados pelas técnicas consideradas segundo a medida IoU no conjunto de teste.	126

Lista de símbolos

A	Matriz de ativações
b	Vetor de <i>bias</i>
B	Número de caixas delimitadoras por célula
C	Número de classes do problema
f	Função
\mathcal{F}	Filtro
\mathcal{J}	Função de custo total
l	Camada da rede neural
L	Número de camadas intermediárias da rede neural
\mathcal{L}	Função de custo
m	Tamanho do conjunto de dados
m'	Tamanho do lote de dados
M	Mapa de características
M	Matriz de mapas de características
\mathcal{M}	Conjunto discreto de épocas
$n^{[l+1]}$	Número de filtros da camada convolucional <i>l</i>
N	Tamanho da imagem
p	Taxa de poda
r	Número de componentes
S	Tamanho da grade de células
S	Matriz de similaridades
T	Número total de épocas
w	Vetor de pesos sinápticos
W	Matriz de pesos sinápticos
\widetilde{W}	Matriz de projeção
x	Vetor de entrada
X	Matriz de entrada
y	Vetor de saída
Y	Matriz de saída
\hat{Y}	Matriz de predições
Z	Matriz de somas ponderadas
α	Taxa de aprendizado
β	<i>Momentum</i>
γ	Fator de redução da taxa de aprendizado
η	Constante de decaimento negativa da taxa de aprendizado
θ	Vetor de parâmetros
κ	Expoente de crescimento da taxa de aprendizado
λ	Peso da regularização
τ	Número de épocas para crescimento da taxa de aprendizado
∇	Gradiente

Sumário

Agradecimentos

Resumo

Abstract

Lista de figuras

Lista de tabelas

Lista de símbolos

1	Introdução	16
1.1	Motivação	18
1.2	Objetivos	20
1.3	Contribuições	21
1.4	Organização do texto	22
2	Trabalhos relacionados	24
2.1	Detecção de fogo	24
2.2	Detecção de fumaça	31
2.3	Conclusões do capítulo	35
3	Referencial teórico	36
3.1	Noções básicas de aprendizado de máquina	36
3.1.1	Algoritmos de aprendizado	37
3.1.2	Validação cruzada	41
3.1.3	Capacidade do modelo	42
3.1.4	Dilema viés e variância	44
3.1.5	Otimização	44
3.2	Redes neurais artificiais	51
3.2.1	Notação matemática	51
3.2.2	Redes neurais profundas	53
3.2.3	<i>Backpropagation</i>	54

3.2.4	Funções de ativação	56
3.2.5	Inicialização de parâmetros	64
3.2.6	Transferência de aprendizado	65
3.2.7	Regularização	66
3.3	Redes neurais convolucionais	70
3.3.1	Camadas convolucionais	71
3.3.2	Camadas de agrupamento	74
3.3.3	Blocos residuais	75
3.3.4	Representação tridimensional	76
3.4	Detecção de objetos	77
3.4.1	Métricas de avaliação de desempenho	77
3.4.2	Detectores baseados em aprendizado profundo	83
3.4.3	<i>You Only Look Once</i> (YOLO)	84
3.5	Conclusões do capítulo	92
4	Metodologia	93
4.1	Base de dados	93
4.1.1	Coleta de imagens	93
4.1.2	Cenários	95
4.1.3	Geração de imagens	97
4.1.4	Rotulação das imagens	99
4.1.5	Estatísticas	100
4.2	Poda de filtros convolucionais	101
4.2.1	Técnicas baseadas em critérios de importância	103
4.2.2	Técnicas baseadas em múltiplas projeções	103
4.2.3	Técnicas baseadas em agrupamento	109
4.3	Conclusões do capítulo	110
5	Resultados	112
5.1	Configuração experimental	112
5.2	Hiperparâmetros	113
5.3	Análise de desempenho	115
5.4	Otimização da rede	120
5.5	Estudo de caso	129
5.6	Conclusões do capítulo	132
6	Conclusões	133
6.1	Trabalhos futuros	134
	Referências bibliográficas	137

Capítulo 1

Introdução

Nos últimos tempos, especialmente desde a década passada, a recorrência de incêndios no mundo vem causando uma enorme preocupação aos cientistas e autoridades de diversos países. Tais eventos causam uma variedade de danos sociais, ambientais e econômicos e não se restringem apenas às áreas de incidência, uma vez que o fogo é capaz de se propagar rapidamente para regiões vizinhas. Dentre os continentes mais afetados estão a África e a América do Sul. A África tem um maior número de incêndios que a América do Sul, entretanto, esses incêndios têm origem em processos menos agressivos, como o cultivo do campo por pequenos agricultores. Já a América do Sul, mesmo com um número menor de eventos, tem uma situação mais preocupante, visto que seus incêndios são fruto de um desmatamento desenfreado [Petesch, 2019].

No caso particular da América do Sul, esses incidentes vêm dificultando inclusive relações comerciais com outros blocos econômicos, como a União Europeia. Durante a estiagem, a frequência alta de incêndios na região central do continente está diretamente associada à Amazônia e ao Complexo do Pantanal. A Amazônia é a maior floresta tropical do mundo e abrange oito países além do Brasil: Bolívia, Colômbia, Equador, Guiana, Guiana Francesa, Peru, Suriname e Venezuela. O Pantanal, por sua vez, é considerado uma das regiões mais úmidas do mundo e abrange Bolívia, Brasil e Paraguai.

De acordo com as estatísticas do satélite AQUA Tarde, da Administração Nacional da Aeronáutica e Espaço (NASA - *National Aeronautics and Space Administration*), cujas medições são a referência utilizada pelo Programa de Queimadas do Instituto Nacional de Pesquisas Espaciais (INPE), estados partes do Mercado Comum do Sul (MERCOSUL) totalizaram 370.729 focos de incêndio apenas no ano de 2020. Os três países com os maiores números de focos de incêndio no continente, isto é, Brasil, Argentina e Bolívia, somaram 73,57% dos incêndios registrados e compõem cerca de 68,49% da extensão territorial da América do Sul [INPE, 2020].

Segundo nota técnica publicada em agosto de 2020 por membros do projeto Ciência para Serviços Climáticos do Brasil (CSSP-BR - *Climate Science for Service*

Partnership Brazil) [Anderson et al., 2020], desde 1988, quando se deu o início do monitoramento de focos de calor por satélites na América do Sul, não haviam sido registrados níveis de queimadas tão altos quanto os obtidos entre março e maio de 2020. O estudo ainda relata que foram identificadas 117 áreas protegidas com probabilidades altas de incêndio nos meses seguintes, isto é, cerca de 433 mil km² em todo o continente sul-americano.

Diante dessas circunstâncias, torna-se essencial implantar estratégias de vigilância para mitigar eventuais prejuízos causados por incêndios ou até mesmo evitá-los. Em grandes extensões territoriais, no entanto, não é factível atribuir tais tarefas a humanos. Além de requerer atenção constante, uma completa varredura da área em busca de possíveis ocorrências de incêndio pode demandar uma mão de obra abundante, custosa e sem garantia de efetividade. Em contraste com essa metodologia e mediante a evolução das técnicas e equipamentos de processamento digital de imagens, o monitoramento assistido por câmeras ganhou grande relevância na gestão ambiental [Jones et al., 2012].

Esse paradigma contemporâneo permite fornecer informações adicionais sobre o incêndio, como a dimensão e a localização através da visualização de imagens, bem como oferece a possibilidade de incorporar aplicações inteligentes a essas câmeras, de modo que as detecções sejam rápidas, assertivas e independam de um agente tomador de decisão. A execução de tarefas complexas por máquinas a partir de uma reprodução, ainda que parcial, da percepção visual humana, como nesse caso, provém da visão computacional (CV - *computer vision*), especialmente sob o foco da inteligência artificial (AI - *artificial intelligence*).

Todavia, elaborar regras determinísticas para orientar o computador no reconhecimento automático de ocorrências de incêndio em uma variedade de cenas e situações sempre foi uma tarefa árdua [Chen et al., 2006; Celik et al., 2007]. Diversos trabalhos nessa perspectiva foram publicados e os de maior relevância consideraram a análise de movimento para mapear as regiões de alta probabilidade de incêndio [Töreyn et al., 2005] e a segmentação nos espaços de cores RGB [Chen et al., 2004], YCbCr [Celik et al., 2007] e YUV [Marbach et al., 2006] para localização dos focos na imagem.

Segundo esses trabalhos, as principais adversidades das estratégias baseadas em modelagem computacional são provenientes de características inerentes ao ambiente, tais como complexidade da cena e condições adversas de iluminação. Em ambientes externos, sobretudo em plantações e florestas, os cenários costumam ser mais complexos e imprevisíveis. Mudanças climáticas, ciclos dia-noite, reflexos de luz, sombras, névoas e luzes artificiais podem provocar variações de luminosidade repentinas e aumentar consideravelmente os níveis de incerteza do problema [Schultze et al., 2006].

Em associação com a visão computacional, o aprendizado profundo (DL - *deep*

learning), um subcampo do aprendizado de máquina (ML - *machine learning*) que, por sua vez, é um subcampo da inteligência artificial, permitiu grandes avanços nas tarefas de reconhecimento e rastreamento de objetos [Krizhevsky et al., 2012; Luo et al., 2020]. Algoritmos de aprendizado profundo são fundamentados em estatística inferencial e, portanto, utilizam ferramentas probabilísticas para tomar decisões na presença de incertezas a partir de um conhecimento adquirido em um grande conjunto de dados. Dentre os modelos dessa classe, o mais adequado para manipular dados não-estruturados, como imagens e vídeos, é a rede neural convolucional (CNN ou ConvNet - *convolutional neural network*), cuja extração de características dos objetos de interesse é realizada diretamente dos pixels brutos da entrada, sem necessidade de pré-processamento ou etapa manual de extração e seleção de características [LeCun et al., 1989b].

A ideia de monitorar eventos através do processamento de dados visuais por redes neurais convolucionais tem crescido vertiginosamente. No âmbito de identificação automática de incêndios, as abordagens de modelagem do fogo e da fumaça por regras discriminativas foram sendo gradativamente substituídas por redes convolucionais treinadas em imagens obtidas por câmeras de vigilância. Os resultados apresentados são bastante promissores e denotam uma boa capacidade de adaptação das arquiteturas de CNNs propostas a ocorrências inéditas de incêndio [Frizzi et al., 2016; Tao et al., 2016; Yin et al., 2017; Muhammad et al., 2018; Govil et al., 2020].

1.1 Motivação

Embora essa tendência por redes convolucionais tenha proporcionado altos níveis de detecção de incêndios, como nunca se havia visto, tais arquiteturas trouxeram consigo alguns desafios. O primeiro deles é a indispensabilidade de um amplo conjunto de imagens contendo as classes de interesse, isto é, fogo e fumaça. Na ausência dessa condição, um algoritmo de aprendizado profundo jamais será capaz de adquirir conhecimento suficiente para identificar incêndios adequadamente em uma imagem ou quadro de vídeo. Entretanto, uma base com muitas imagens pode ainda não ser o suficiente. Os exemplos têm que ser substancialmente representativos, ou seja, devem abranger eventos de fogo e fumaça em suas mais variadas formas e posições, bem como cenários reais em condições adversas. De outro modo, falsos alarmes serão emitidos recorrentemente, promovendo descredibilidade do sistema e ainda causando esforços desnecessários para combater incêndios que não existiram.

Infelizmente, até o desenvolvimento dessa pesquisa, não havia bases de dados diversificadas com um elevado número de exemplos disponíveis gratuitamente na lite-

ratura, principalmente para a tarefa de detecção, onde as imagens devem ser fornecidas juntamente com os rótulos dos objetos, indicando a qual classe eles pertencem, e com as coordenadas das caixas que compreendem esses objetos, indicando suas respectivas localizações na imagem. Essa carência de dados rotulados eventualmente advém do trabalho exaustivo que é coletar milhares de imagens e, posteriormente, marcar manualmente, com caixas delimitadoras, a localização de todos os focos de incêndio compreendidos em cada uma dessas imagens.

Outro dilema das CNNs é a relação de compromisso (*trade-off*) que elas assumem entre desempenho e velocidade de execução [Huang et al., 2017]. À medida que as arquiteturas das redes convolucionais se tornam mais complexas, geralmente melhores são seus desempenhos em tarefas de reconhecimento de objetos [Dhingra, 2017]. Contudo, a complexidade dessas topologias está intimamente relacionada ao aumento do número de parâmetros, o que pode implicar, em muitos casos, no aumento das operações de ponto flutuante (FLOPs) efetuadas e, conseqüentemente, no tempo necessário para processar cada quadro de vídeo.

Dado que as redes convolucionais profundas consomem bastante memória e demandam um elevado custo computacional, tanto no treinamento quanto na predição, sua implantação em dispositivos com recursos limitados, como *smartphones* e computadores de placa única (Raspberry Pi [Raspberry Pi Foundation, 2021], BeagleBoard [Texas Instruments, 2021], Tinker Board [AsusTek Computer Inc., 2021] e similares) se torna inviável. Para lidar com esses problemas, diversos estudos vêm sendo conduzidos com o propósito de otimizar as arquiteturas e implementações dessas redes. As principais frentes investigadas incluem binarização de parâmetros [Rastegari et al., 2016; Zhu e Liu, 2020], convolução separável em profundidade [Howard et al., 2017; Sandler et al., 2018], poda de estruturas [Li et al., 2016; Anwar et al., 2017; Jordao et al., 2020a,b] e destilação do conhecimento (*knowledge distillation*) [Buciluă et al., 2006; Hinton et al., 2015; Chen et al., 2017]. As abordagens de poda, no entanto, compreendem simultaneamente os benefícios de todas as técnicas mencionadas, tais como redução de memória e diminuição do número de parâmetros, sendo então as mais investigadas pela comunidade científica atualmente.

Filtros convolucionais são essencialmente matrizes de parâmetros que, quando aplicados a uma imagem, geram uma nova imagem através da operação de convolução. As imagens resultantes então possibilitam enfatizar diferentes informações contidas na imagem original. Em contraste com os métodos primitivos, onde os filtros eram modelados manualmente por especialistas, nas redes convolucionais esses filtros são aprendidos durante o processo de treinamento e são dispostos em camadas convolucionais [Goodfellow et al., 2016]. Entretanto, os filtros convolucionais aprendidos podem

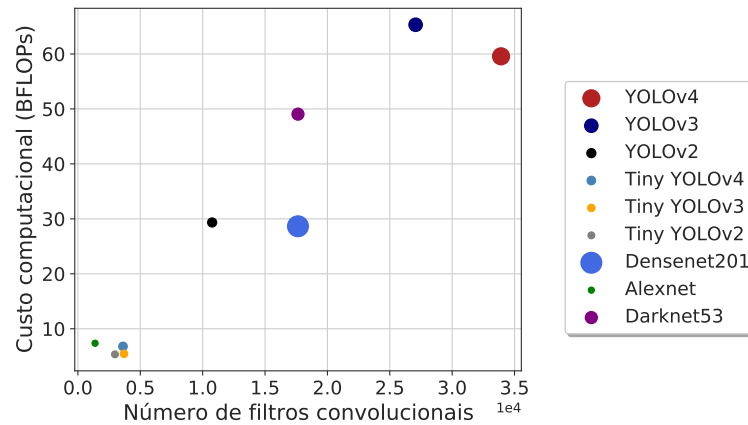


Figura 1.1: Relação de compromisso entre custo computacional e número de filtros convolucionais de algumas CNNs projetadas para tarefas de classificação e detecção. O custo computacional é mensurado pelo número de operações de ponto flutuante executadas por propagação direta da rede em uma imagem RGB de dimensão 416×416 pixels e o tamanho de cada ponto indica a profundidade da arquitetura em termos do número de camadas convolucionais.

apresentar uma redundância significativa em relação a quais características do objeto que devem ser identificadas na imagem ou podem até mesmo pouco contribuir para discriminação das classes de interesse. Uma vertente de poda de CNNs atua justamente na remoção desses filtros menos relevantes, tal que se tenha uma redução imediata no custo computacional durante a predição sem redução da profundidade da arquitetura. A Figura 1.1 mostra a relação de compromisso entre o número de filtros convolucionais e o custo computacional de algumas CNNs comumente encontradas na literatura.

Em harmonia com a necessidade de se monitorar grandes áreas verdes com múltiplas câmeras integradas a sistemas de detecção de incêndios de baixo custo e com processamento local, a poda de filtros pode atuar como solução ao gargalo de custo computacional dos sistemas baseados em redes convolucionais profundas. Assim, a partir de uma rede convolucional de arquitetura complexa treinada para indicar a presença de fogo e fumaça na imagem, é possível otimizá-la para operar em dispositivos móveis e *hardwares* com baixo poder computacional.

1.2 Objetivos

O presente trabalho tem como objetivo geral elaborar uma ferramenta autônoma e eficiente de detecção de incêndios computacionalmente factível para dispositivos com recursos limitados. Os objetivos específicos podem ser enumerados conforme o ciclo de vida tradicional de um algoritmo de aprendizado de máquina juntamente com uma

fase de otimização do modelo:

1. coleta de imagens¹ contendo ocorrências de incêndio em diversas fontes de informação gratuitas, visando qualidade, variedade e quantidade, uma vez que isso implica em grande parcela do sucesso da rede neural convolucional e, consequentemente, a sua robustez a incertezas;
2. rotulação manual, por meio de caixas delimitadoras, de todo fogo e fumaça presentes nas imagens coletadas;
3. construção do modelo de detecção de objetos para a localização de fogo e fumaça em imagens e quadros de vídeo a partir do treinamento com os dados coletados;
4. ajuste de hiperparâmetros para maximizar o desempenho do modelo e o seu potencial de generalização para situações de incêndio inéditas;
5. otimização do modelo através de técnicas de poda de filtros convolucionais para melhor funcionamento em dispositivos móveis, bem como máquinas desprovidas de unidades de processamento gráfico.

1.3 Contribuições

Mediante a escassez de grandes conjuntos de dados rotulados para a tarefa de detecção de incêndios, a primeira contribuição dessa pesquisa foi justamente a concepção de uma base de dados (intitulada D-Fire) que, até o momento, compreende mais de 21.000 imagens, sendo cerca de 11.000 delas com ocorrências reais de incêndio. As imagens e rótulos contendo as localizações das classes fogo e fumaça estão disponíveis gratuitamente para treinamento de modelos de aprendizado de máquina em geral, assim como para diversos outros estudos relacionados ao tema de identificação de incêndios [Gaia, 2018].

Outra contribuição relevante é a elaboração de um modelo robusto de detecção de fogo e fumaça a partir desses dados. Por se tratar de um modelo de aprendizado profundo, cuja arquitetura contém milhões de parâmetros, o processamento dos quadros de vídeo só se torna computacionalmente factível com o suporte de unidades de processamento gráfico (GPUs - *graphics processing units*). Tal requisito, no entanto, pode gerar um alto custo de infraestrutura para implementação em grandes áreas de monitoramento.

¹Todas as imagens coletadas são coloridas, com informação apenas na faixa visível do espectro eletromagnético.

Em vista disso, a última e principal contribuição desse trabalho é otimizar a arquitetura do modelo de detecção de incêndios desenvolvido, de modo que as detecções possam ser realizadas em dispositivos mais baratos, isto é, reduzidos em processamento, memória e consumo de energia elétrica. Para esse fim, foi realizada uma investigação do efeito de diferentes técnicas de podas de filtros convolucionais no detector de incêndios proposto. Os principais trabalhos de redução de parâmetros em redes convolucionais disponíveis na literatura se referem tipicamente à tarefa de classificação de imagens, ou seja, quando o objetivo é atribuir um único rótulo de classe por cena, sem ainda qualquer informação de localização. Assim, ainda não existem muitas análises similares para a tarefa de detecção de objetos [Ghosh et al., 2019], especialmente para o caso particular da detecção de incêndios. Finalmente, um estudo de caso apresenta simulações do modelo final em um Raspberry Pi 4 [Raspberry Pi Foundation, 2021].

1.4 Organização do texto

Segundo uma breve contextualização da alta frequência de queimadas nos principais biomas da América do Sul, o Capítulo 1 relata uma notável tendência por soluções de combate a incêndios baseadas em inteligência artificial, descreve os principais desafios para tornar tais sistemas robustos e escaláveis em grandes áreas verdes e aponta as principais motivações que originaram a solução proposta para contornar esses problemas. No Capítulo 2 tem-se uma revisão da literatura acerca das técnicas de detecção de fogo e fumaça das últimas duas décadas, as principais adversidades encontradas ao longo dos anos, o que já foi solucionado e o que ainda resta a ser resolvido pelos métodos mais recentes.

O Capítulo 3 retrata a fundamentação teórica necessária para a compreensão desse trabalho, desde as noções básicas de aprendizado de máquina até a construção da arquitetura complexa do detector de objetos considerado. O principal objetivo nesse caso é garantir uma leitura clara e concisa de como os modelos atuais funcionam e onde são feitas as otimizações em suas arquiteturas para garantir menor custo computacional e, por consequência, maior velocidade de processamento. De maneira complementar, o Capítulo 4 detalha o processo de aquisição de imagens para composição da base de dados D-Fire, bem como as técnicas investigadas para remoção dos filtros convolucionais.

As decisões tomadas para construção do modelo de detecção de fogo e fumaça, os experimentos conduzidos para validação das diferentes técnicas de podas de redes convolucionais nesse cenário, as discussões dos resultados obtidos e o estudo de caso

são apresentados no Capítulo 5. Por fim, o Capítulo 6 fornece as últimas considerações por meio de conclusões, limitações da abordagem proposta e direções promissoras para pesquisas futuras em diferentes vertentes do tema.

Capítulo 2

Trabalhos relacionados

Este capítulo relata o histórico do progresso das técnicas de detecção de fogo e fumaça desenvolvidas nas últimas duas décadas e, a partir das limitações atuais, apresenta os principais motivos que fomentaram a condução do presente trabalho.

2.1 Detecção de fogo

Dentre os sistemas desenvolvidos para detecção de incêndio, os mais convencionais são baseados em sensores térmicos, fotoelétricos ou iônicos. No entanto, suas primeiras aplicações de sucesso ficaram restritas a ambientes fechados, visto que esses dispositivos apresentam alcance limitado e ainda exigem determinada proximidade com o foco de incêndio.

Inicialmente, pesquisas referentes a essas abordagens surgiram com o intuito de investigar suas limitações através das incertezas presentes na modelagem dos sensores. Yuen e Chow [2005] propuseram uma abordagem probabilística no projeto de um sistema térmico de detecção de incêndios, onde o método de Monte Carlo [Metropolis e Ulam, 1949] foi utilizado para estudar os efeitos das incertezas associadas à localização e expansão do fogo, além de estabelecer uma análise estatística da eficácia do projeto e das possíveis alternativas.

No contexto de segurança, uma abordagem é considerada confiável quando o tempo necessário para a detecção e supressão de um incêndio é menor que o tempo para que situações de perigo ocorram. Dessa forma, Joglar et al. [2005] desenvolveram um modelo probabilístico para prever a ativação de um detector de teto em diferentes cenários de incêndio, considerando ainda o impacto das incertezas presentes nos parâmetros responsáveis por seu desempenho, como taxa de liberação de calor, temperatura de ativação, tempo de ativação e localização do dispositivo.

A escalabilidade para áreas abertas e extensas tornou-se possível mediante o paradigma de redes de sensores sem fio, cujo princípio é implantar um conjunto de nós sensores na região de interesse para monitorar e reportar possíveis ocorrências de

incêndio. A arquitetura contém um ou mais nós sorvedouros responsáveis por coletar todas as informações dos nós sensores e transmitir a uma estação base, que provê comunicação direta com os usuários [Raghavendra et al., 2006].

Sob essa perspectiva, Yu et al. [2005] apresentaram um método de detecção de incêndio em tempo real utilizando redes de sensores sem fio. Os nós sensores foram implantados em uma floresta para coletar dados climáticos e enviar aos nós sorvedouros. Esses processavam os dados através de uma rede neural responsável por calcular a probabilidade de incêndio a partir das condições climáticas atuais. De posse desses fatores, um nó mestre concluía a taxa de risco de incêndio florestal. Em caso de situações emergenciais, como mudanças repentinas na temperatura ou presença de fumaça, os próprios nós sensores enviavam diretamente um alerta ao nó mestre.

Singh e Singh [2012] também utilizaram uma rede de sensores sem fio. Entretanto, a detecção do fogo foi realizada por lógica difusa do tipo 2 [Karnik e Mendel, 1998], visto que os dados coletados pelos nós sensores podem gerar incertezas na tomada de decisão. Por exemplo, não se sabe com exatidão quanto de cada fator climático deve estar envolvido para que ocorra um incêndio. Além de garantir maior precisão nos resultados através da redução significativa da taxa de falsos alarmes, o sistema proposto demandou menos energia, processamento e memória, fatores cruciais para a durabilidade de uma rede de sensores sem fio.

Em virtude da necessidade de corroborar a detecção do incêndio através da visualização de imagens, carência notável nas técnicas até então desenvolvidas, Lloret et al. [2009] propuseram uma abordagem híbrida de redes de sensores sem fio com câmeras IP (*internet protocol*). Assim como nos métodos anteriores, o sensor responsável pela identificação do incêndio alertava uma estação base. No entanto, esta também era encarregada de selecionar a câmera mais próxima da ocorrência, posicioná-la corretamente e enviar a ela uma mensagem para aquisição instantânea das imagens. Ainda que fosse fundamental a análise das cenas por um profissional, a integração da câmera com a rede sem fio diminuiu a incidência de falsos alarmes e manteve o tempo de resposta aceitável.

Apesar dos métodos baseados em múltiplos sensores apresentarem resultados promissores, dimensioná-los para áreas externas requer bastante investimento e diligência. Geralmente, regiões florestais compõem biomas cuja cobertura vegetal é densa, com árvores de grande porte contíguas. Assegurar um posicionamento adequado dos sensores, nessas condições, se torna inviável, visto que o acesso durante a instalação e manutenção dos equipamentos é restrito. Além disso, a extensão da área de monitoramento afeta diretamente o custo do projeto e fragiliza ainda mais a gestão dos recursos disponíveis para o funcionamento da rede.

Diante das dificuldades evidenciadas e de uma constante evolução tecnológica em câmeras digitais e computadores, diversos pesquisadores se interessaram por estudos na área de visão computacional para monitorar eventos em imagens. Collins et al. [2000] apresentaram um sistema de vigilância capaz de analisar objetos a partir de um vídeo, determinar suas localizações geográficas e classificá-los em categorias semânticas (humanos, carros e caminhões). Analogamente, Boulton et al. [2001] exploraram, porém em um domínio de alta relevância militar, a detecção de alvos não cooperativos e camuflados em ambientes externos.

Independentemente do panorama do monitoramento, todas soluções convergem para intervenções de cunho preventivo. Entretanto, particularizar os eventos como ocorrências de incêndio sempre foi desafiador, principalmente através de uma vertente da visão computacional conhecida como *top-down*, cujo objetivo é descrever regras para identificação de objetos em suas diversas formas e posições [Borenstein et al., 2004].

Na tentativa de identificar chamas em quadros de vídeos, Cheng et al. [1999] relataram investigações nas diferenças de luminância entre o fogo e outros objetos utilizando imagens em escala de cinza. Durante os experimentos, foi observado que objetos refletivos podem possuir luminância semelhante à de objetos emissores em algumas circunstâncias. Mesmo com a utilização de um filtro na câmera para discriminar os objetos pelo espectro no infravermelho, ainda houve falsas detecções. Contudo, os autores propuseram uma análise de pixels ao longo do tempo para avaliar as áreas de incidência e expansão do fogo.

Chen et al. [2004] estabeleceram um método de detecção baseado no espaço de cores RGB (*red, green, blue* - vermelho, verde, azul). A principal regra utilizada na medição cromática ($R \geq G > B$) analisava especialmente a intensidade e saturação da componente R para extrair potenciais pixels de fogo. Posteriormente, a taxa de crescimento desses pixels era investigada em sequências de quadros para validar o incêndio, assim como a persistência espaço-temporal empregada no trabalho de Cheng et al. [1999].

A segmentação por cores promoveu grande progresso na modelagem do fogo e foi devidamente explorada em diversas representações. Marbach et al. [2006], por exemplo, propuseram um modelo para reconhecer incêndios em quadros de vídeo através da luminância e cintilância dos pixels no espaço de cores YUV, onde Y é a componente de luminância e U e V são, respectivamente, as componentes azul e vermelha de crominância. Os padrões identificados tratam a frequência de oscilação do fogo na faixa de 1 a 10 Hz e a luminância dos pixels associados ao fogo próxima do valor máximo permitido pela câmera (nível de saturação).

Schultze et al. [2006] também constataram que as frequências de oscilação me-

didadas não excederam 10 Hz. Em análises de movimento realizadas com câmeras de alta velocidade, uma similaridade entre a frequência de cintilância dos quadros e do som foi verificada, elucidando que a oscilação das chamas pode ser mensurável tanto visual quanto acusticamente. Porém, de acordo com os autores, deve-se tomar outras considerações para garantir uma detecção robusta, visto que alguns objetos em ambientes externos, como reflexos de luz em superfícies de água, apresentam propriedades semelhantes e podem causar falsos alarmes.

Segundo as conclusões de diversas pesquisas sobre as dificuldades de se encontrar características que sejam intrínsecas e exclusivas do fogo, Celik et al. [2007] propuseram um detector de incêndios que combina três estratégias: (i) subtração de fundo; (ii) classificação de pixels no espaço RGB; e (iii) variações temporais. A subtração de fundo, responsável por detectar objetos dinâmicos em cenas com fundos estáticos, foi adaptada com o auxílio de distribuições gaussianas [Stauffer e Grimson, 1999] para também tratar mudanças nos cenários. Um modelo genérico de cores foi construído após uma análise estática das imagens contendo pixels de fogo e permitiu adicionar regras de discriminação entre luminância e crominância ao trabalho de Chen et al. [2004]. Finalmente, as variações em grupos de pixels no decorrer dos quadros permitiu avaliar se havia expansão das áreas candidatas.

Ainda que a associação de estratégias para definição de regras tenha aumentado substancialmente a taxa de detecção, aplicações referentes ao contato imediato com a brigada de incêndio ou acionamento de quaisquer medidas preventivas são consideradas praticáveis apenas mediante um baixo índice de falsos alarmes. Caso contrário, um número expressivo implicaria, por exemplo, em mobilizações desnecessárias para contenção de um fogo inexistente várias vezes ao dia, bem como de uma desconfiança bem grande por parte dos usuários da aplicação.

Cientes dessas circunstâncias, Celik e Demirel [2009] atribuíram seus esforços ao aperfeiçoamento do classificador de pixels de fogo, pressupondo que esse seja o principal gargalo para se obter um algoritmo robusto a falsos positivos. As novas regras, fundamentadas nos estudos de Chen et al. [2004] e Celik et al. [2007], utilizaram o espaço de cores YCbCr para uma melhor separação entre as componentes dos sinais de vídeo ($Y > Cb$ e $Cr > Cb$), onde Y é a componente de luminância e Cb e Cr são as componentes de crominância azul e vermelha, respectivamente. Apesar da redução na taxa de falsos alarmes de 60% para 31,5%, Celik [2010] estendeu as pesquisas para a modelagem no espaço de cores CIE (*Commission Internationale de L'Eclairage* - Comissão Internacional de Iluminação) L^*a^*b em busca de melhores resultados, onde L^* indica luminância e a^* e b^* representam as quatro cores exclusivas da visão humana: vermelho, verde, azul e amarelo.

Durante o esforço dos pesquisadores para equiparar as detecções automáticas às obtidas pela visão humana, isto é, para identificar apenas os reais eventos de incêndio, modelos que tomam decisões em dados de entrada por meio de um conjunto de regras e exceções se mostraram frágeis na tentativa de representar a complexidade do fogo e de suas respectivas variações de iluminação, posição e forma. Por essa razão, a vertente *bottom-up* da visão computacional, ainda recente, adquiriu visibilidade em diversos domínios [Borenstein et al., 2004].

Assim como o cérebro humano é treinado para inúmeras tarefas, a máquina pode aprender a reconhecer padrões nos dados que lhe são apresentados e, posteriormente, usar o conhecimento adquirido para interpretar e analisar dados desconhecidos. As redes neurais artificiais, um dos mais versáteis algoritmos de aprendizado de máquina, tiveram seu primeiro marco de pesquisa na década de 40 e, mesmo diante de grandes progressos em suas arquiteturas, foram desconsideradas por um período conhecido como inverno da inteligência artificial (1987-1993) [Russell e Norvig, 2002]. Foi a ideia certa no momento errado. Eram necessários um grande volume de dados, bem como memória para armazená-los e poder computacional para processá-los [Data Science Academy, 2019].

Nos anos 2000, ocorreu o que ficou conhecido como tempestade perfeita, onde todos os fatores imprescindíveis para a utilização das redes neurais em problemas complexos foram conquistados. Unidades de processamento gráfico para realização de operações matriciais em alta velocidade se tornaram realidade e uma era de produção desenfreada de dados por instituições e pessoas, denominada *big data*, se estabeleceu [Pias et al., 2019].

Angayarkkani e Radhakrishnan [2010] aproveitaram o cenário favorável e utilizaram dados espaciais provenientes de técnicas de sensoriamento remoto, isto é, imagens de satélite, como entrada da metodologia descrita a seguir. As imagens foram convertidas previamente para o espaço de cores CIE XYZ, onde Y é a luminância e X e Z correspondem, respectivamente, ao tom e à saturação provenientes da cromaticidade. Em seguida, as imagens transformadas foram segmentadas por difusão anisotrópica para identificar as prováveis regiões de incêndio. Os valores dos pixels pertencentes a essas regiões foram então utilizados para treinar uma rede neural com função de ativação de base radial (RBF - *radial basis function*). Embora a eficácia tenha sido comprovada em vários bancos de dados públicos, imagens de satélites não apresentam visualização contínua e, portanto, impossibilitam uma rápida detecção.

No trabalho de Cheng et al. [2011], a rede RBF também foi empregada para detecção de incêndios, porém seu treinamento foi realizado com dados sensoriais, como temperatura do fogo e concentração de monóxido de carbono no ambiente. O pro-

cesso de aprendizado foi executado em duas fases: (i) aprendizado não supervisionado para definir os parâmetros das funções de base radial (centro e dispersão) através do algoritmo de agrupamento K-médias; e (ii) aprendizado supervisionado para treinar os pesos da camada de saída através do método do gradiente. Além da rápida convergência, a simulação apresentou probabilidades médias de erro inferiores a 5% na identificação dos três estados (fogo, fogo latente, ausência de fogo).

As primeiras redes neurais a serem exploradas no âmbito de vigilância ambiental, como a rede RBF citada e a de alimentação direta (*feedforward*), dispõem de apenas uma camada de processamento, também conhecida como camada oculta ou intermediária. Embora estas arquiteturas fossem bastante simples, seu uso permitiu que várias tarefas de visão computacional fossem resolvidas com precisão. Todavia, múltiplas camadas ocultas podem conceber aprendizados muito mais ricos [He et al., 2016]. Em uma tarefa de reconhecimento de padrões visuais, por exemplo, a primeira camada poderia ser responsável por aprender bordas, a segunda camada por aprender formas geométricas construídas a partir dessas bordas e assim sucessivamente.

À medida que a rede se torna mais profunda, isto é, apresenta mais camadas ocultas, maiores também são os desafios que circundam seu processo de treinamento [Srivastava et al., 2015]. Em redes neurais cujos neurônios de cada camada estão conectados com todos os neurônios das camadas adjacentes (redes totalmente conectadas), o aprendizado fica ainda mais lento em meio ao número expressivo de operações matemáticas. Suponha que em um problema de classificação de imagens, a imagem de entrada é colorida (3 canais de cores) e de tamanho 128×128 pixels. Como cada pixel da imagem está ligado a cada neurônio da camada de entrada da rede, as camadas totalmente conectadas estabelecem 49.152 pesos apenas na primeira camada intermediária. Porém, esse não é o único problema. As camadas podem aprender em velocidades distintas devido à instabilidade intrínseca associada ao aprendizado baseado em gradiente (VGP - *vanishing gradient problem*) [Hochreiter et al., 2001]. Quando pesos são inicializados aleatoriamente, algumas camadas ficam presas a mínimos locais, interferindo na generalização do modelo [Bengio, 2009].

Diferentemente das redes clássicas, as rede neurais convolucionais, uma arquitetura elaborada anos antes por Fukushima [1980] e utilizada por LeCun [1989] para reconhecer dígitos manuscritos (LeNet), viabilizam o gerenciamento de imagens através de uma filtragem de conexões baseada em proximidade. Para evitar que cada neurônio seja associado a cada pixel de entrada, as conexões se tornam restritas a uma pequena parte da imagem, na qual cada neurônio só realiza conexões até um determinado número de pixels de distância. Mesmo que uma imagem contenha milhares ou milhões de pixels, é possível detectar características relevantes através de filtros (*kernels*) que

ocupam apenas dezenas de pixels na imagem [Goodfellow et al., 2016]. Essa operação, conhecida como convolução, ocorre nas camadas convolucionais e só é possível sem que haja desperdício de informações porque normalmente a correlação dos pixels diminui à medida que a distância entre eles aumenta.

Em dicotomia com as demais topologias que exigem recursos projetados manualmente por especialistas, as redes convolucionais aprendem a representação de características a partir dos pixels brutos da imagem. Dessa forma, modelagem em espaços de cores e persistência espaço-temporal não são mais fundamentais no reconhecimento de objetos, mesmo que ainda possam ser complementares. Zhang et al. [2016] projetaram uma rede convolucional para extrair as características visuais relevantes dos dados e utilizá-las no treinamento de dois classificadores. A detecção ocorreu em cascata, onde o classificador global foi responsável por verificar se há fogo na imagem e o classificador de granulação fina por identificar, quando existente, a localização estimada da ocorrência. Diante da escassez de dados rotulados na comunidade acadêmica para o problema em questão, um conjunto de dados com 25 vídeos foi elaborado para avaliação do método proposto, cuja precisão foi de 90% nos dados de teste.

Shi et al. [2017] também realizaram uma metodologia híbrida e sequencial. A princípio, o algoritmo de detecção de saliência PISA (*Pixelwise Image Saliency by Aggregating*) proposto por Wang et al. [2015] foi empregado para extrair regiões candidatas a apresentar fogo. Essas regiões foram utilizadas para treinar a AlexNet, uma rede convolucional com oito camadas (cinco convolucionais e três totalmente conectadas) projetada por Krizhevsky et al. [2012] e premiada no Desafio de Reconhecimento Visual de Grande Escala do ImageNet (ILSVRC ou ImageNet - *ImageNet Large Scale Visual Recognition Challenge*) em 2012. Os experimentos realizados em uma base de imagens coletada pelos próprios autores corroboraram a superioridade das redes convolucionais em comparação com os algoritmos determinísticos modelados nos espaços de cores RGB e YCbCr.

Sem quaisquer etapas de pré-processamento e engenharia de recursos (*feature engineering*), a solução apresentada por Muhammad et al. [2018] integrou um circuito fechado de televisão (CFTV) com uma rede convolucional para assegurar um monitoramento autônomo e robusto. Ao utilizarem uma arquitetura com cinco camadas de convolução e três camadas totalmente conectadas, foi possível obter detecções de incêndio ainda em estágios iniciais e uma taxa de falsos positivos inferior a 9,5%. Em outras métricas de avaliação comumente utilizadas, como precisão, revocação, acurácia e medida F_1 , a rede foi capaz de superar, em duas bases de dados distintas, dez métodos clássicos da literatura, entre eles os de Chen et al. [2004] e Celik e Demirel [2009].

2.2 Detecção de fumaça

Apesar de o fogo ser a principal analogia visual do incêndio, detectá-lo nem sempre garante supressão instantânea. Dependendo da situação, como quando as ocorrências estão muito distantes ou existem obstáculos impossibilitando uma visão clara, encontrar o fogo é extremamente difícil e, quando possível, ele só é identificado após um período em que já foram atingidas proporções suficientes para grandes prejuízos.

Uma oportunidade verossímil de se evitar a propagação ampla e descontrolada de incêndios é também considerar a fumaça para detecção. Geralmente, uma ocorrência de incêndio é precedida por um acúmulo de gás em sua origem resultante da combustão, o que permite utilizar esse evento como indicador de alerta em qualquer circunstância. No entanto, sensores de fumaça comuns apresentam limitação em relação ao tempo de resposta, visto que as partículas de carbono podem demorar para atingir o detector de ponto, causando um fenômeno conhecido como atraso de transporte [Töreyn, 2018].

Sistemas baseados em vídeos podem ser aplicados em condições que os métodos sensoriais falham. Câmeras monitoram longas distâncias e grandes espaços abertos, além de não apresentarem os atrasos de transporte e de limiar térmico. Logo que a fumaça se faz presente no ângulo de visão do sistema, é possível advertir imediatamente o corpo de bombeiros para que sejam tomadas as providências cabíveis. Por consequência, utilizar câmeras para identificar precocemente a fumaça, conforme inferido também para o fogo, torna-se uma possibilidade bem mais interessante [Çetin et al., 2013].

Para um ser humano, realizar a detecção de fumaça é uma tarefa relativamente trivial. Porém, replicar tal capacidade para as máquinas já se torna um grande desafio, uma vez que a fumaça não possui forma definida e pode ser facilmente confundida com neblinas e nuvens. Em busca de regras representativas, Kopilovic et al. [2000] investigaram irregularidades no movimento da fumaça mediante o cálculo do fluxo óptico [Barron et al., 1994] em dois quadros consecutivos, no qual a entropia das distribuições de velocidade foi considerada para diferenciar a fumaça de objetos com rigidez.

Guillemant e Vicente [2001] propuseram um método de inclusão temporal capaz de extrair, em pequenos envelopes espaciais, movimentos similares aos de fumaça, isto é, de pequenos objetos deformáveis e dinâmicos. A partir da análise cumulativa dos dados de movimento instantâneo, os autores distinguiram a fumaça de reflexos e de árvores estremecidas por ventos e nuvens. A energia da distribuição de velocidade no envelope para a fumaça foi superior à energia da maioria desses fenômenos, com exceção das nuvens, que foram singularizadas por um desvio padrão de distribuição menor. Ainda que alguns objetos em movimento tenham sido contrastados com a fumaça e suas

respectivas propriedades espaço-temporais, muitos objetos predominantes em cenas de vigilância ambiental não foram analisados.

A dinâmica da fumaça é bastante característica e, por isso, foi explorada em vários outros estudos. Normalmente, são observados movimentos laterais e ascendentes, cuja área de propagação se expande ao longo do tempo e a velocidade se altera de acordo com a intensidade das chamas [Friendlander, 2000]. Todavia, muito do que é observado depende claramente do referencial da câmera, o que implica em diversas exceções nos padrões convenientemente assumidos.

A par desses comportamentos, Töreyn et al. [2005] criaram uma abordagem com múltiplas etapas para reconhecimento de fumaça admitindo câmeras estáticas. A primeira técnica consistiu em mapear as regiões de movimento por subtração de fundo. Em seguida, a transformada wavelet foi aplicada para uma análise no conteúdo de alta frequência da imagem, onde observou-se que uma diminuição dos valores dos extremos locais era um indicador de fumaça. Outra constatação foi que a presença da fumaça implica em uma redução da crominância dos pixels nos canais U e V do espaço de cores YUV, dado que o meio se torna mais cinzento. Ainda foram verificados comportamentos periódicos nos limites da fumaça e convexidade das regiões para a classificação final.

Não satisfeitos com os falsos alarmes obtidos em situações de ambiente noturno, os autores aprimoraram o trabalho em duas outras oportunidades, sendo uma com câmeras tradicionais [Töreyn et al., 2006] e outra com câmeras infravermelho [Töreyn et al., 2007]. As análises de tremulação e movimento da fumaça no domínio wavelet, bem como a subtração de fundo, continuaram em vigor. Entretanto, o critério de energia, obtido a partir da soma das componentes espaciais de alta frequência, foi adotado e a dinâmica de sua variação foi obtida por um modelo oculto de Markov (HMM - *hidden Markov model*) treinado com pixels de fumaça e de objetos análogos.

Com o propósito de antecipar os alertas promovidos pelo detector de fogo apresentado em Chen et al. [2004], Chen et al. [2006] também propuseram uma estratégia baseada em regras conjuntas para classificação de pixels de fumaça. As decisões foram sustentadas por uma análise estática e por uma análise dinâmica. A primeira consistiu em avaliar os níveis de cinza (claro e escuro) no espaço de cores RGB através da cromaticidade, tal como seus respectivos limites na componente I do espaço HSI (*hue, saturation and intensity* - matiz, saturação e intensidade). A segunda, por sua vez, constituiu-se de uma investigação do processo de difusão da fumaça, em que são consideradas as taxas de crescimento e de desordem. Os resultados experimentais evidenciaram que o método cumpriu com a proposta de aviso prévio, porém com falsos positivos em locais de baixa luminosidade.

A tendência notável das redes neurais artificiais como estado da arte para diversas tarefas também inspirou metodologias associadas ao aprendizado da fumaça. No princípio, as redes neurais eram frequentemente utilizadas como classificadores incorporados às estratégias baseadas em regras. Chunyu et al. [2010], por exemplo, utilizaram estimativa de fundo, regras de decisão nos espaços de cores RGB e HSI e recursos dinâmicos por fluxo óptico para determinar as regiões candidatas a fumaça. Por fim, uma rede neural treinada com *backpropagation* [Rumelhart et al., 1986] foi utilizada para discriminar os recursos entre fumaça e não-fumaça. Apesar dos resultados serem dependentes dos valores estatísticos considerados para treinamento da rede, o método proposto superou o de Töreyn et al. [2006], principalmente para fumaças com tonalidades mais escuras.

De maneira similar, Yu et al. [2013] utilizaram um modelo de cores baseado nos espaços RGB e HSI para avaliar regiões de movimento definidas por subtração de fundo. As regiões prováveis a apresentarem fumaça ainda foram submetidas ao processamento de imagem em bloco e ao cálculo do fluxo óptico. Os recursos extraídos foram utilizados no treinamento de uma rede neural artificial para classificação da fumaça. Embora os resultados apresentados tenham gerado boas perspectivas, os autores avaliaram o método apenas em 9 vídeos, número insuficiente para garantir robustez em cenários com variações periódicas de iluminação.

Em um segundo momento, avanços teóricos na área de aprendizado de máquina, maior disponibilidade de recursos computacionais e uma quantidade massiva de dados viabilizaram investimentos de inúmeros pesquisadores nas redes neurais convolucionais. Dado o alto desempenho obtido por essa topologia no campo da visão computacional, Hohberg [2015] verificou que o uso de CNNs para detecção de fumaça em incêndios florestais é uma abordagem muito promissora para aumentar o desempenho de sistemas autônomos baseados em câmeras. Ainda que a proposta do trabalho não tenha sido elaborar uma arquitetura própria, modelos treinados foram capazes de detectar 88% das regiões com fumaça.

Frizzi et al. [2016] apresentaram uma rede neural convolucional para classificação de incêndios composta por quatro camadas convolucionais e duas camadas totalmente conectadas. Testada em 5.584 imagens extraídas de vídeos reais, o que corresponde a 20% da base de dados considerada, a arquitetura adotada realizou detecções de múltiplas classes, isto é, tanto de fogo quanto de fumaça, e alcançou uma acurácia média de 97,9%.

Redes convolucionais profundas, no geral, assumem um compromisso entre desempenho e velocidade de execução [Girshick, 2015; Redmon et al., 2016]. Adicionar camadas intermediárias em uma rede permite que recursos cada vez mais abstratos

sejam aprendidos e usados em sequência, bem como também possibilita uma representação mais fácil das interações nos dados. No entanto, quanto maior a profundidade de uma rede, maior a complexidade computacional nas fases de treinamento e predição. Assim, redes mais profundas costumam apresentar melhor desempenho e, consequentemente, menor velocidade de execução.

Colocando em prova tal conflito, de forma a encontrar a melhor configuração possível para detecção de fumaças, Tao et al. [2016] desenvolveram uma rede convolucional cuja arquitetura apresentava cinco camadas convolucionais e três camadas totalmente conectadas. O treinamento foi realizado em duas etapas independentes: (i) considerando uma base de dados pequena (1.383 imagens) e (ii) considerando uma base de dados maior (10.712 imagens). Em conjuntos de teste com proporções similares aos dos conjuntos de treinamento, isto é, 1.505 imagens e 10.617 imagens, o modelo submetido a menor partição de dados obteve uma taxa de detecção de 94,2% e uma taxa de falsos alarmes de 0,86% e o segundo modelo obteve taxas de 99,4% e 0,44%, respectivamente.

Por meio de modificações mais expressivas na estrutura das redes convolucionais, Yin et al. [2017] propuseram uma rede composta por 14 camadas, onde todas as camadas convolucionais eram acompanhadas de normalização em lote [Ioffe e Szegedy, 2015] para acelerar o processo de treinamento e aumentar a precisão da identificação de fumaça. Os autores ainda tiveram a cautela de tratar o sobreajuste causado por amostras desequilibradas e insuficientes, expandindo o conjunto de treinamento a partir das imagens originais usando várias técnicas de expansão de dados (*data augmentation*) [Mikołajczyk e Grochowski, 2018]. A rede desenvolvida apresentou simultaneamente baixa taxa de falsos alarmes e alta taxa de detecção e também dominou dois métodos baseados em descritores de textura.

Outra abordagem interessante para suprir a escassez de dados foi a elaboração de imagens de fumaça sintéticas por Zhang et al. [2018]. As imagens foram produzidas inserindo-se fumaças reais e simuladas no fundo de fotografias com paisagens predominantemente florestais. Ao treinar o detector de objetos baseado em região *Faster* R-CNN [Ren et al., 2016] com esses dados e depois testá-lo em imagens de incêndios reais, foi possível concluir que modelos treinados por imagens sintéticas de fumaça também são eficazes na prática, mesmo que as imagens não sejam visualmente realísticas.

As redes convolucionais profundas superam efetivamente as redes rasas (*shallow learning*) para detecção de incêndios [Hohberg, 2015]. Entretanto, aplicar o estado da arte em ambientes com grau de incerteza elevado não implica necessariamente em bons resultados. As CNNs apenas atingem desempenhos satisfatórios e rapidez durante o monitoramento, podendo ultrapassar inclusive a capacidade humana, quando sustentadas por GPUs e bem projetadas sob um grande volume de exemplos. Diante do

exposto, o presente trabalho propõe soluções para ambos requisitos: (i) um conjunto de imagens rotuladas para treinamento de redes de detecção de fogo e fumaça e (ii) técnicas para otimização de redes treinadas nesse conjunto de imagens, viabilizando assim a execução desses sistemas em dispositivos com baixa capacidade computacional.

Conforme mencionado, a técnica de otimização de CNNs selecionada foi a poda de filtros convolucionais. Apesar dos resultados notáveis com esse tipo de otimização [Li et al., 2016; Hu et al., 2016; Luo e Wu, 2017; Jordao et al., 2020b], identificar quais filtros devem ser removidos é um campo que ainda vem sendo bastante explorado, uma vez que remover filtros indiscriminadamente pode causar uma alta depreciação no desempenho da rede neural [Liu et al., 2020]. Em consequência disso, uma série de ideias vêm sendo testadas para mensurar a importância dos filtros em cada uma das camadas convolucionais das redes. As técnicas disponíveis na literatura vão desde simples métricas aplicadas diretamente nos filtros [Li et al., 2016] a propostas mais elaboradas, que também buscam analisar correlações entre filtros [Ghosh et al., 2019] e interações com a saída da rede [Jordao et al., 2020b]. Algumas técnicas de cada uma dessas vertentes serão descritas na Seção 4.2.

2.3 Conclusões do capítulo

Este capítulo apresentou um resumo dos principais sistemas de detecção de incêndios da literatura, desde os sensores de fumaça iônicos tipicamente instalados em residências até as soluções de monitoramento ambiental de larga escala baseadas em visão computacional. Dentre as técnicas utilizadas para reconhecimento de padrões de fogo e fumaça, as que se mostram mais promissoras atualmente são baseadas em redes neurais convolucionais. Contudo, isso vem ao custo de modelos complexos com uso intensivo de memória e computação, sugerindo potenciais contribuições nesses aspectos.

Capítulo 3

Referencial teórico

Este capítulo fornece o referencial teórico necessário para a compreensão da metodologia empregada e das contribuições realizadas no presente trabalho. A construção dos conceitos ocorre gradualmente, isto é, dos princípios básicos até os mais avançados, de modo que a leitura seja clara tanto para especialistas quanto para iniciantes no tema.

3.1 Noções básicas de aprendizado de máquina

Desde o princípio da computação, pesquisadores do mundo inteiro se questionavam sobre a possibilidade de máquinas se tornarem inteligentes. No entanto, até meados da década de 60, para que fosse viável automatizar uma tarefa humana, ainda era essencial programar explicitamente instruções em um computador. Esse processo manual de modelagem da entrada a partir de regras para se obter a saída desejada, conhecido como paradigma tradicional da programação, foi se tornando ineficiente à medida que as tarefas de interesse atingiam níveis de abstração e complexidade cada vez mais altos.

A inteligência artificial, por sua vez, mostrou ser capaz de resolver tarefas naturalmente executadas por humanos mas difíceis de se descrever formalmente, isto é, problemas resolvidos intuitivamente no cotidiano e que são complicados de se codificar, como reconhecimento de fala, identificação de objetos em uma cena e até mesmo condução de automóveis [Goodfellow et al., 2016]. Todavia, o sucesso da área não emergiu mediante uma trajetória completamente triunfante. Vários projetos buscaram codificar o conhecimento em regras de inferência lógica, porém sem muito êxito. As dificuldades enfrentadas ao longo dos anos indicaram que os sistemas baseados em inteligência deveriam adquirir seu próprio conhecimento a partir dos dados. Foi então que uma subárea da inteligência artificial denominada aprendizado de máquina surgiu.

O termo aprendizado de máquina se popularizou em 1959 com um estudo de Arthur Lee Samuel, engenheiro da IBM (*International Business Machines Corpora-*

tion) e pioneiro em jogos de computador baseados em auto-aprendizado. De acordo com Arthur Samuel, a essência desse novo paradigma é proporcionar às máquinas a capacidade de aprender sem que haja a programação explícita [Samuel, 1959]. Em outras palavras, o processo de aprendizado é construído segundo a observação dos exemplos fornecidos e o conhecimento adquirido é utilizado em tomadas de decisão futuras.

3.1.1 Algoritmos de aprendizado

Em conformidade com a concepção de que os algoritmos de aprendizado de máquina dispõem de alto potencial adaptativo, Tom Michael Mitchell define concisamente a percepção abstrata de aprendizado nesse contexto:

Diz-se que um programa de computador aprende com a experiência E em relação a alguma classe de tarefas T e medida de desempenho P , se seu desempenho nas tarefas em T , medido por P , melhorar com a experiência E [Mitchell, 1997].

Tal afirmação permite assimilar dois principais aspectos: (i) aprender é justamente o mecanismo que proporciona a competência de desempenhar a tarefa; ii) a construção do algoritmo de aprendizado depende notoriamente de inúmeros fatores, onde se pode imaginar uma variedade de categorias para cada um.

3.1.1.1 Experiência E

A experiência E consiste no conhecimento propriamente dito e nas particularidades para sua aquisição. O princípio, obviamente, provém de uma coleção de exemplos preestabelecida (*dataset*), afinal, é necessário extrair o aprendizado de algum lugar. Essa relação particular com os dados e de como eles estão disponíveis para uso, no entanto, fornece vertentes distintas de aprendizado. As principais são:

1. Aprendizado supervisionado: nesse âmbito, a supervisão refere-se à oportunidade que o algoritmo tem de, durante o aprendizado, verificar a resposta desejada para qualquer exemplo disponível e, assim, validar se o caminho assumido é promissor. Dessa forma, cada exemplo do conjunto de dados deve estar associado à sua verdade fundamental (*ground-truth*), também conhecida como rótulo (*label*). Suponha que seja do interesse de um indivíduo identificar a raça dos animais por uma simples fotografia. É provável que o algoritmo tenha dificuldades em distinguir um husky siberiano de um lobo cinzento, porém se lhe for informado as raças de todos os exemplos, ele terá capacidade de aprimorar seu conhecimento

a partir dos seus próprios erros. Esse rótulo, no entanto, não é de concepção trivial, sendo geralmente definido por especialistas.

2. Aprendizado não-supervisionado: em contrapartida, na aprendizagem não-supervisionada, os rótulos não são exigidos. O algoritmo tem como princípio aprender a distribuição de probabilidades que originou os dados por meio de uma extração de características úteis e representativas. Então por que não optar sempre pelo aprendizado não-supervisionado? Na realidade, cada vertente de aprendizado compreende tipos de tarefas específicas. Em problemas de agrupamento (*clustering*), por exemplo, a proposta não requer rótulos, mas sim organizar o conjunto de dados em grupos de acordo com as similaridades identificadas.

Embora esses paradigmas se apresentem com conceitos complementares, é possível se deparar com situações intermediárias, onde há um conjunto de dados com apenas alguns exemplos rotulados. Intuitivamente, é comum querer descartá-los para tratar o problema como aprendizado não-supervisionado. Contudo, pode ser que extrair recursos ou características (*features*) relevantes dos dados seja difícil, o que torna esses exemplos rotulados valiosos. Então por que não rotular os demais? Pode ser uma possibilidade quando se tem tempo e mão de obra especializada. Caso contrário, uma alternativa mais factível é utilizar os dados como estão, conforme sugere o aprendizado semi-supervisionado. Assim, aprendizados restritos aos exemplos sem rótulos podem ser aperfeiçoados [Van Engelen e Hoos, 2020].

Por fim, alguns algoritmos de aprendizado não adquirem sua experiência E em bases de dados predeterminadas. O aprendizado por reforço (*reinforcement learning*) propõe uma interação com um ambiente condicionado às diversas incertezas e o conhecimento provém dos erros e acertos nas tomadas de decisão. Para que a tarefa seja concluída, toda ação promovida pelo algoritmo é ponderada por recompensas e penalizações [Mnih et al., 2013].

3.1.1.2 Tarefa T

No que se refere às tarefas T , tudo é estabelecido em conciliação com o modo de processamento de um único exemplo. E um exemplo é, essencialmente, um conjunto de características mensuradas quantitativamente, seja de um objeto ou de um evento que se deseja investigar. Tipicamente, a comunidade representa um exemplo i como um vetor $\mathbf{x}^{(i)} \in \mathbb{R}^n$, no qual cada entrada x_{ij} do vetor representa uma característica j do exemplo i [Goodfellow et al., 2016]. Em visão computacional, onde um exemplo

normalmente é uma imagem ou um quadro de vídeo, as características podem ser tanto os valores brutos dos pixels quanto alguma transformação deles.

Diversos tipos de tarefas podem ser resolvidos por aplicações que contemplam tecnologias de aprendizado e esse escopo vem se expandindo substancialmente com a evolução da área e do poder computacional. Entretanto, duas das tarefas em aprendizado supervisionado são mais relevantes para o presente trabalho:

1. **Regressão:** nesse tipo de tarefa, o algoritmo aprende a mapear n características de um exemplo de entrada em um valor de saída contínuo através de uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$. As três categorias mais comuns de regressão são: (i) regressão linear simples, cujo objetivo é estimar a relação entre uma variável independente e outra variável dependente por meio de uma reta; (ii) regressão linear múltipla, uma extensão da anterior para duas ou mais variáveis independentes; e (iii) regressão polinomial, onde deseja-se encontrar uma relação não linear entre variáveis independentes e dependentes. Exemplos clássicos de problemas de regressão são precificação dinâmica e previsão de demanda.
2. **Classificação:** é um tipo de tarefa similar à regressão, exceto pelo formato da saída, que é restrita a um conjunto de C possíveis categorias. Desse modo, o algoritmo prevê um rótulo discreto para as variáveis de entrada a partir de uma função $f : \mathbb{R}^n \rightarrow \{1, \dots, C\}$. Quando o problema compreende apenas duas classes ($C = 2$), tal como a presença ou não de uma doença em pacientes, tem-se um caso especial denominado classificação binária. No caso de um número maior de classes ($C \geq 3$), por exemplo, reconhecimento de dígitos manuscritos, o problema é dito classificação multiclasse.

Além desses dois tipos de tarefas, muitos outros são possíveis de se solucionar com aprendizado de máquina, como detecção de anomalias [Mehrotra et al., 2017], tradução automática [Koehn, 2009], segmentação semântica [Hao et al., 2020] e detecção de objetos [Jiang et al., 2019]. Esse último será introduzido na Seção 3.4 como uma combinação de múltiplos problemas.

3.1.1.3 Medida de desempenho P

Um algoritmo de inteligência ao final de sua rotina de aprendizado produz um modelo com várias variáveis de configuração interna conhecidas como parâmetros. Uma analogia bastante utilizada na literatura clássica de aprendizado de máquina aborda o modelo como uma hipótese e os seus parâmetros como a adaptação da hipótese a um conjunto de dados específico [Mitchell, 1997].

A fim de avaliar a experiência E adquirida, métricas quantitativas foram desenvolvidas para cada tipo de tarefa T . A verdadeira intenção da avaliação é prever qual será o desempenho do modelo no mundo real antes de implantá-lo. Nessa perspectiva, emprega-se um procedimento de divisão dos dados (*data splitting*), cuja razão central é simular cenários semelhantes aos ambientes de desenvolvimento, homologação e produção de um sistema.

A estratégia mais comum propõe uma separação aleatória dos dados em três conjuntos independentes e identicamente distribuídos (i.i.d) [Reitermanová, 2010]:

1. Conjunto de treinamento: amostra de dados utilizada para estimar os parâmetros do modelo. É a partir dessas observações que haverá a aquisição de conhecimento para execução da tarefa de interesse. As medidas de desempenho utilizadas sobre o conjunto de treinamento servem para analisar a convergência do algoritmo, ou seja, a qualidade de ajuste dos parâmetros do modelo a esses dados.
2. Conjunto de validação: do mesmo modo que um modelo possui variáveis de configuração interna, ele tipicamente também possui variáveis de configuração externa. Tais variáveis, intituladas como hiperparâmetros, contribuem na estimativa dos parâmetros e não podem ser aprendidas a partir dos dados, sendo, portanto, definidas por heurísticas ou empiricamente. Geralmente, o conjunto de validação é opcional e provém do conjunto de treinamento. Seu principal propósito é fornecer uma avaliação imparcial de desempenho, que auxiliará tanto na escolha da melhor configuração de hiperparâmetros quanto na constatação da capacidade do modelo mediante dados desconhecidos.
3. Conjunto de teste: finalmente, o modelo cujos hiperparâmetros produziram a melhor medida de desempenho no conjunto de validação é avaliado no conjunto de teste. Essa amostra tem como propósito assegurar que o modelo está preparado para ser implantado no ambiente de produção, bem como proporcionar, quando necessário, uma comparação final não tendenciosa entre diferentes modelos.

A priori, é normal assimilar os conjuntos de validação e teste como redundantes. No entanto, eles não são. Uma vez que há um empenho significativo no ajuste dos hiperparâmetros para promover um aprendizado mais robusto dos parâmetros, é possível que o modelo fique enviesado, se tornando apto apenas para situações similares às encontradas nos conjuntos de treinamento e validação.

3.1.2 Validação cruzada

O processo de divisão dos dados pode ser um problema quando se tem uma base de dados relativamente pequena. Tal impasse surge da proporção de dados que será designada para os conjuntos de treinamento e teste, principalmente se do conjunto de treinamento ainda forem selecionadas algumas observações para composição de um conjunto de validação. Na hipótese de um conjunto de teste com um número significativo de observações, é provável que o conjunto de treinamento não seja suficiente para viabilizar um aprendizado robusto. Em contrapartida, um conjunto de teste menor implica em uma estimativa ruim do desempenho do modelo, onde a métrica empregada será altamente enviesada pelas poucas observações disponíveis para avaliação.

Alternativas para esse cenário priorizam uma proporção maior para o conjunto de treinamento, de modo a garantir um aprendizado adequado, bem como avaliam o desempenho do modelo a partir da média de múltiplas estimativas. Esses procedimentos são baseados no princípio da repetição, onde o modelo é treinado e avaliado em várias divisões arbitrárias da base de dados, minimizando a incerteza estatística intrínseca ao processo aleatório de composição dos conjuntos.

Dentre esses procedimentos, o mais utilizado é a validação cruzada *k-fold*. Suponha uma base de dados dividida em dois conjuntos i.i.d, onde $(100-p)\%$ das observações constituem o conjunto de treinamento e $p\%$ constituem o conjunto de teste [Goodfellow et al., 2016]. A partir do conjunto de treinamento são originadas k partições independentes, na qual $\frac{1}{k}$ são destinadas à otimização de hiperparâmetros e $\frac{k-1}{k}$ ao aprendizado dos parâmetros. Assim, a estimativa de desempenho de uma determinada configuração do modelo pode ser obtida pelos valores médios da métrica de avaliação no subconjunto de validação em k experimentos (Figura 3.1). Por fim, os exemplos reservados para o conjunto de teste são utilizados para uma avaliação imparcial do melhor modelo definido, simulando a performance real em um ambiente de produção.

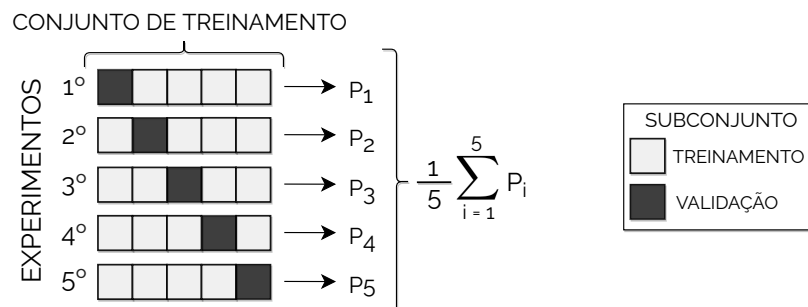


Figura 3.1: Exemplo de validação cruzada *k-fold* com $k = 5$. No experimento i , o i -ésimo subconjunto é utilizado para validação e os demais para treinamento. O modelo de maior média da medida de desempenho P nos 5 experimentos é tido como o melhor.

3.1.3 Capacidade do modelo

Um dos principais desafios de se construir um modelo de aprendizado de máquina é torná-lo apto a cumprir com o seu propósito em entradas inéditas, isto é, observações nunca vistas durante os processos de aprendizado e ajuste de hiperparâmetros. Essa habilidade de apresentar um bom desempenho em exemplos desconhecidos é denominada generalização [Goodfellow et al., 2016].

Assim como a convergência é avaliada sobre o aprendizado do algoritmo no conjunto de treinamento, a generalização pode ser mensurada sobre as saídas produzidas pelo modelo no conjunto de teste. No aprendizado supervisionado, onde se tem as saídas desejadas, é possível quantificar o erro entre o que se obteve e o que se espera como resposta. A comparação feita considerando os dados de treinamento é designada de erro de treinamento e essa mesma comparação nos dados de teste configura o erro de generalização, também conhecido como erro de teste. Em divisões que consideram um conjunto exclusivo para ajuste de hiperparâmetros do modelo, tem-se como métrica de referência um erro de validação.

Sob essa perspectiva, pode-se dizer que um bom modelo é aquele que satisfaz dois principais requisitos: (i) erro de treinamento pequeno e (ii) diferença absoluta entre os erros de treinamento e teste (*gap*) também pequena [Goodfellow et al., 2016]. No que tange a violação do primeiro requisito, tem-se um modelo de baixa capacidade, cujo ajuste ao conjunto de treinamento não foi bem-sucedido. Em situações como essa, é dito que o modelo sofreu um subajuste (*underfitting*). A violação do segundo requisito, por sua vez, implica em um modelo de alta capacidade, no qual foi excessivamente ajustado para os exemplos do conjunto de treinamento e, portanto, não generaliza bem para novas observações. Nesse caso, o modelo sofreu um sobreajuste (*overfitting*).

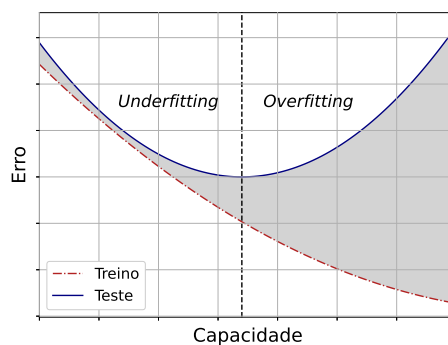


Figura 3.2: Relação entre a capacidade do modelo e os erros de treinamento e teste. A área hachurada corresponde ao *gap* entre os erros e a linha vertical tracejada indica o momento estimado em que o modelo atinge sua capacidade ótima, também conhecido como *early stopping point*.

Para que um modelo se ajuste adequadamente aos dados de treinamento e ainda apresente um alto potencial de generalização, sua capacidade deve estar em sintonia com a real complexidade da tarefa que lhe foi designada (Figura 3.2). Essa necessidade se torna ainda mais evidente quando se investiga modelos com capacidades variadas em problemas clássicos da literatura, como apresentado nas Figuras 3.3 e 3.4.

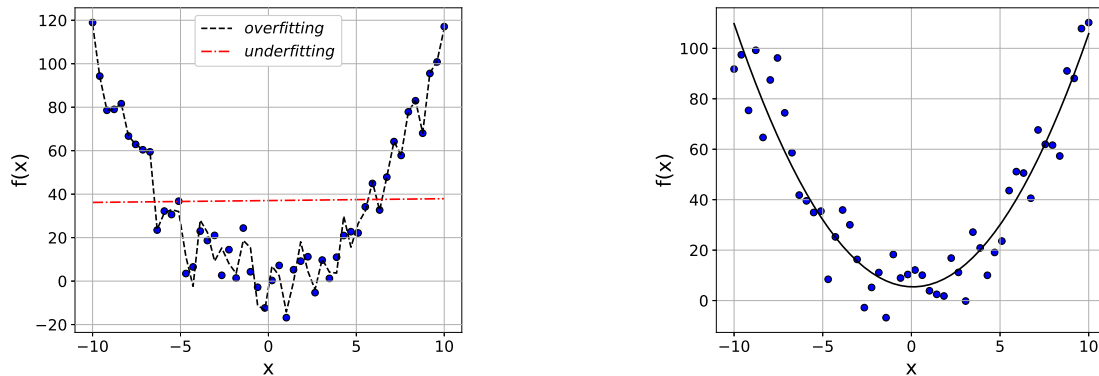


Figura 3.3: Dado um problema de regressão polinomial, cuja função que se deseja aproximar é uma parábola, considere três modelos para solução do problema: um polinômio de grau 1, um polinômio de grau 2 e um polinômio de grau 50. O polinômio de grau 1 é muito simples e, portanto, não tem capacidade suficiente para capturar a curvatura da função. No entanto, modelos complexos demais também podem não ser ideais, como é o caso da aproximação ruidosa do polinômio de grau 50 (à esquerda). Um bom ajuste pode ser visto à direita, onde a aproximação apresenta um aspecto mais suave e fidedigno da função quadrática.

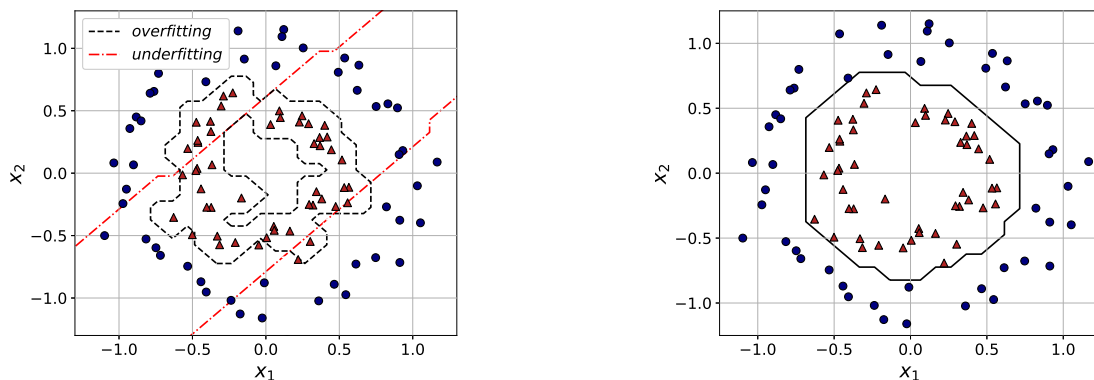


Figura 3.4: Em um problema de classificação binária onde as classes estão dispostas em círculos concêntricos, por exemplo, um modelo demasiadamente simples não será capaz de realizar o contorno de separação entre as observações de classes distintas. Analogamente, um modelo de alta complexidade produzirá uma superfície de separação bastante específica para o conjunto no qual foi treinado e, por consequência, não se adequará a dados novos ligeiramente distintos dos aprendidos (à esquerda). O melhor modelo para o problema é capaz de prover a superfície de decisão mais simples que separa as categorias existentes, como é apresentado à direita.

3.1.4 Dilema viés e variância

O erro de generalização pode ser decomposto em erro de viés (*bias error*) e erro de variância (*variance error*). O primeiro termo trata basicamente da diferença entre a saída obtida e a saída desejada. Já o segundo termo consiste na sensibilidade do modelo a pequenas flutuações provenientes de uma amostra particular dos dados de treinamento. Em processos de geração de dados inerentemente estocásticos, tem-se ainda o erro de *Bayes*, também conhecido como erro irreduzível (*irreducible error*), que representa o ruído referente ao próprio problema e, portanto, não pode ser eliminado com a escolha do algoritmo [Friedman, 1997].

A capacidade ideal de um modelo está altamente associada ao equilíbrio entre as componentes viés e variância que promove a minimização do erro total. Assim, qualquer composição que destoe dessa harmonia resultará em perda de performance. Um modelo com alto viés tende a ter uma estrutura subjacente mais simples e é um típico exemplo de *underfitting*, pois denota rigidez no reconhecimento de padrões relevantes nos dados. Em contraste, um modelo com alta variância normalmente dispõe de uma estrutura subjacente mais complexa e apresenta *overfitting*, visto que não generaliza para novos cenários. Essa relação de compromisso foi intitulada de dilema viés e variância [Geman et al., 1992].

3.1.5 Otimização

A otimização trata-se de um processo de decisão que envolve encontrar a melhor solução \mathbf{s}^* de um problema dentre um conjunto Ω de soluções factíveis [Chong e Zak, 2004]. Matematicamente, a otimização pode ser definida como a minimização ou maximização de uma função objetivo $f(\mathbf{s}) : \mathbb{R}^n \rightarrow \mathbb{R}$ por meio de mudanças nas variáveis de decisão $\mathbf{s} \in \mathbb{R}^n$.

Problemas de otimização são geralmente formulados em termos da minimização da função objetivo, portanto é comum que se faça a conversão do problema de maximização de $f(\mathbf{s})$ em minimização de $-f(\mathbf{s})$. Desse modo, convencionou-se sem perda de generalização que a melhor solução será o mínimo global $\mathbf{s}^* = \arg \min f(\mathbf{s})$, ou seja, a solução \mathbf{s} que acarretará no menor valor possível da função objetivo.

Embora o mínimo global seja a meta pretendida em qualquer problema de minimização, nem sempre é possível encontrá-lo. Isso se deve ao fato de que funções objetivo minimamente relevantes na prática apresentam vários pontos críticos como dificultadores (Figura 3.5). Todavia, mínimos locais que mapeiam valores substancialmente baixos da função são frequentemente admissíveis como soluções, especialmente em problemas não triviais e de alta dimensionalidade.

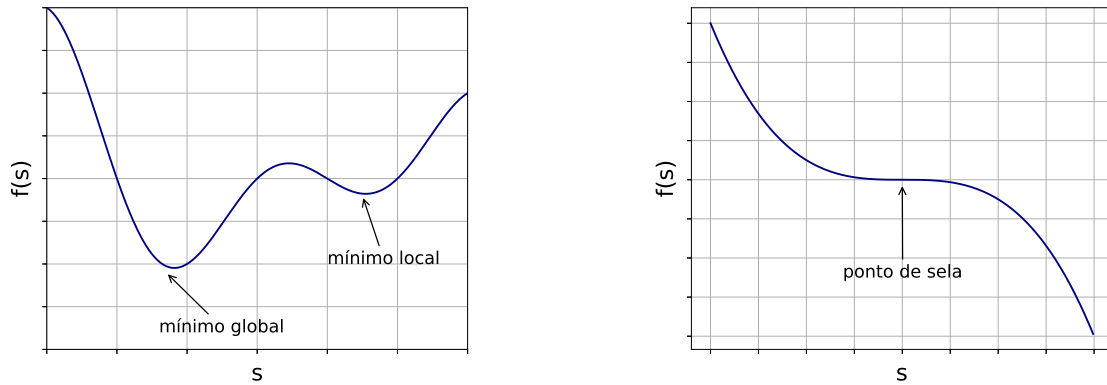


Figura 3.5: Pontos críticos podem ser mínimos locais, ou seja, valores onde a função $f(\mathbf{s})$ é menor apenas em uma dada vizinhança e pontos de sela, que apresentam declividade nula e se situam próximos à regiões tanto com direção maximizante quanto com direção minimizante.

No âmbito do aprendizado de máquina, os inúmeros parâmetros de um modelo são usualmente estimados através de otimização. Durante o aprendizado, esse processo busca valores dos parâmetros que minimizem o erro de treinamento e, conseqüentemente, elevem o desempenho do modelo na tarefa desejada. Nesse caso, a função objetivo, cuja finalidade é avaliar a qualidade de cada conjunto selecionado de parâmetros, assume diferentes nomenclaturas, tais como função de perda, função de custo e função de erro [Goodfellow et al., 2016]. No presente trabalho todos esses termos serão utilizados intercambiavelmente com a notação $\mathcal{L}(\boldsymbol{\theta})$, onde $\boldsymbol{\theta}$ são os parâmetros do modelo.

3.1.5.1 Gradiente descendente

Um dos algoritmos de otimização mais tradicionais é o gradiente descendente (GD - *gradient descent*) [Cauchy, 1847]. Como o próprio nome já sugere, o método se baseia no cálculo do gradiente $\nabla\mathcal{L}(\boldsymbol{\theta})$, vetor perpendicular à curva de nível que passa pelo ponto $\boldsymbol{\theta}$ e que indica a direção e o sentido de maior crescimento da função objetivo $\mathcal{L}(\boldsymbol{\theta})$, que deve ser diferenciável. A fim de encontrar uma solução ótima, o gradiente descendente considera, a cada iteração i , um passo positivo α (taxa de aprendizado) na direção oposta à direção do gradiente, isto é, na direção minimizante e, em seguida, atualiza os valores das variáveis de decisão de acordo com:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \alpha \nabla\mathcal{L}(\boldsymbol{\theta}_{i-1}) \quad (3.1)$$

A convergência desse método numérico é dada quando a norma dos n elementos do gradiente é menor ou igual a um valor de tolerância ϵ consideravelmente pequeno,

isto é, $\|\nabla\mathcal{L}(\boldsymbol{\theta}_i)\| < \epsilon$. Como pontos críticos podem satisfazer tal condição, é possível que o algoritmo fique preso em regiões de mínimos locais.

3.1.5.2 Gradiente descendente em lote

Algoritmos de aprendizado de máquina requerem um vasto conjunto de dados para adquirirem uma boa generalização. Contudo, processar um número muito grande de exemplos pode ser computacionalmente caro, uma vez que a função de perda total $\mathcal{J}(\boldsymbol{\theta}_i)$ em uma iteração i é normalmente calculada como a média dos custos $\mathcal{L}(\boldsymbol{\theta}_i)$ para cada exemplo k no conjunto de treinamento de tamanho m :

$$\mathcal{J}(\boldsymbol{\theta}_i) = \frac{1}{m} \sum_{k=1}^m \mathcal{L}_k(\boldsymbol{\theta}_i) \quad (3.2)$$

assim, ao utilizar o gradiente descendente, o custo computacional para estimar a média dos gradientes $\nabla\mathcal{J}(\boldsymbol{\theta}_i)$ numericamente a cada iteração é $\mathcal{O}(m)$ e, portanto, cresce linearmente com o número de observações de treinamento [Goodfellow et al., 2016].

Perante esse problema do custo proibitivo, uma extensão do gradiente descendente denominada gradiente descendente em lote (*mini-batch gradient descent*) foi proposta [Bottou, 1998]. A intuição desse método consiste essencialmente na utilização de pequenos lotes (*mini-batches*) de m' observações a cada iteração para estimar o gradiente, de modo a tornar a complexidade computacional da atualização do gradiente em cada iteração constante $\mathcal{O}(m')$ e independente do tamanho m .

Em outros termos, uma amostra \mathbb{B} de m' observações é extraída aleatoriamente do conjunto de treinamento e o gradiente é calculado apenas sobre esses exemplos no presente instante. Em vista disso, a atualização das variáveis de decisão é ligeiramente modificada conforme:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \alpha \nabla\mathcal{J}(\boldsymbol{\theta}_{i-1}) = \boldsymbol{\theta}_{i-1} - \alpha \left(\frac{1}{m'} \sum_{k=1}^{m'} \nabla\mathcal{L}_k(\boldsymbol{\theta}_{i-1}) \right) \quad (3.3)$$

tal que m' é fixo e substancialmente pequeno. No caso particular de $m' = 1$ tem-se o gradiente descendente estocástico (SGD - *stochastic gradient descent*), bem como para $m' = m$ tem-se a versão clássica do gradiente descendente. A escolha do tamanho m' então deve ser conduzida cautelosamente, assim como qualquer outro hiperparâmetro. Além disso, o número de iterações necessárias para amostrar m dados do conjunto de treinamento é conhecido como época.

Na prática, a estimativa não enviesada do gradiente por meio de uma amostra consideravelmente pequena pode influenciar no percurso de minimização, já que a atua-

lização das variáveis de decisão favorece localmente os exemplos contidos no lote atual. Por consequência, a descida deixa de ser suave e começa a demonstrar maior instabilidade, mas que na média ainda converge para o mínimo local, conforme exemplificado na Figura 3.6 [Goodfellow et al., 2016].

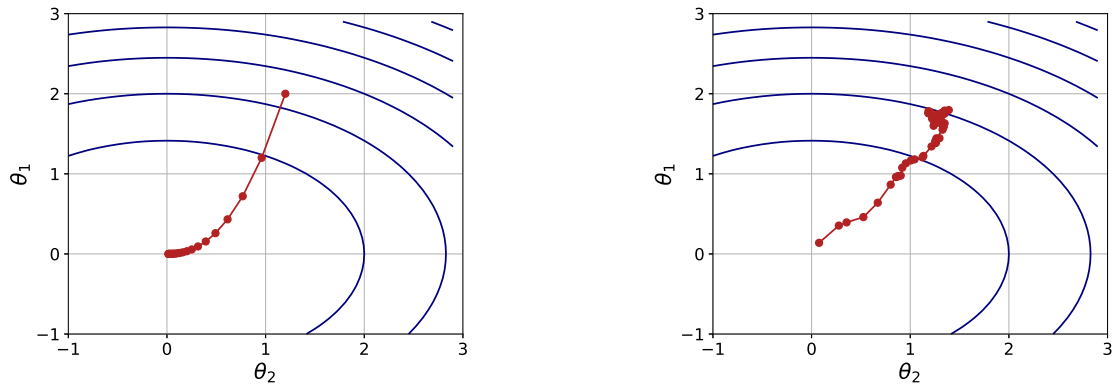


Figura 3.6: A trajetória iterativa do gradiente descendente em lote com $m' = 2$ (à direita) na minimização de $f(\boldsymbol{\theta}) = \theta_1^2 + 4\theta_2^2$ apresenta um comportamento mais oscilatório do que a do gradiente descendente (à esquerda) para a mesma função e mesma taxa de aprendizado $\alpha = 0,1$. Durante a composição do *mini-batch*, a amostragem de exemplos uniformemente distribuída pode arbitrariamente escolher um número maior de exemplos ruidosos e não representativos, o que pode ocasionar pequenos desvios na direção do vetor gradiente em uma dada iteração.

3.1.5.3 Momentum

Em alguns casos, a oscilação durante a otimização pelo método do gradiente descendente em lote pode fazer com que a variável independente ultrapasse a solução ótima até divergir. Intuitivamente, o primeiro procedimento a se adotar é a diminuição da taxa de aprendizado α , de tal forma que os passos não sejam grandes o suficiente para saltarem o mínimo. Entretanto, uma taxa de aprendizado muito pequena pode reduzir drasticamente a velocidade de convergência do algoritmo e, assim, tornar ineficaz o treinamento de modelos com milhares de parâmetros.

Com o intuito de acelerar o aprendizado, principalmente diante de funções objetivo com curvaturas acentuadas e gradientes de baixas magnitudes, o algoritmo do impulso (*momentum algorithm*) [Polyak, 1964] foi adaptado ao contexto dos métodos de otimização baseados em gradiente. Fundamentalmente, essa estratégia direciona o gradiente corrente em conformidade com a média móvel exponencial de gradientes passados, o que permite suavizar sua trajetória de descida.

Para tal fim, assume-se nesse método um vetor de velocidade \mathbf{v}_i definido como a soma de todos os gradientes antecessores ponderada pela taxa de aprendizado e no

qual se pode expressar recursivamente através de:

$$\mathbf{v}_i = \beta \mathbf{v}_{i-1} + \alpha \nabla \mathcal{J}(\boldsymbol{\theta}_{i-1}) \quad (3.4)$$

onde o hiperparâmetro *momentum* $\beta \in [0, 1]$ determina com que rapidez é o decaimento das contribuições dos gradientes anteriores. Um β grande resultará em uma velocidade dependente dos gradientes mais antigos, enquanto que um valor de β pequeno influenciará a velocidade com os gradientes mais recentes. Caso β seja nulo, tem-se o gradiente descendente em lote descrito na Subseção 3.1.5.2.

A atualização das variáveis passa então a depender do módulo e da direção de uma sequência de gradientes, sendo dada por:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \mathbf{v}_i \quad (3.5)$$

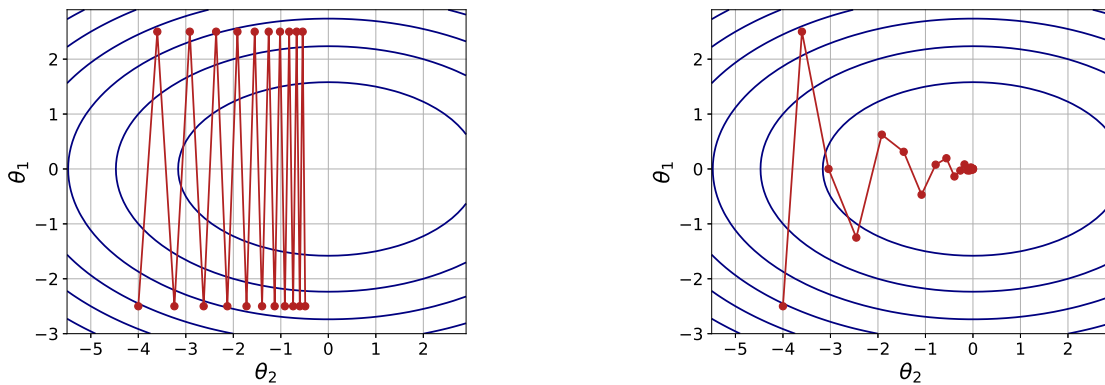


Figura 3.7: Dada a mesma função objetivo da Figura 3.6 e uma taxa de aprendizado cinco vezes maior ($\alpha = 0,5$), tem-se um comportamento completamente instável na minimização pelo gradiente descendente em lote (à esquerda). Em contrapartida, ao adicionar o *momentum* com $\beta = 0,5$ ao método, é possível perceber uma convergência mais rápida e com clara suavização da trajetória.

3.1.5.4 Taxa de aprendizado variável

Conforme discutido anteriormente, a performance do modelo é muito sensível à taxa de aprendizado e, portanto, isso a torna um dos hiperparâmetros mais relevantes no treinamento de modelos de aprendizado de máquina por métodos baseados em gradiente. Em virtude disso, algumas outras alternativas foram elaboradas para conciliar uma trajetória mais estável (comum em situações com α pequeno) com uma convergência mais rápida (comum em situações com α grande).

Dentre essas estratégias, tem-se os agendadores da taxa de aprendizado (*learning rate schedulers*). Em síntese, a taxa de aprendizado é definida inicialmente com um valor grande, tal que os primeiros passos em direção ao mínimo sejam propositalmente maiores e acelerem o aprendizado. À medida que o treinamento decorre e consequentemente os parâmetros se aproximam de uma bacia de atração, a taxa de aprendizado é reduzida, evitando que a solução ótima local seja ultrapassada.

O decaimento da taxa de aprendizado, no entanto, pode apresentar diversos comportamentos ao longo das épocas (Figura 3.8). O *multistep scheduler* [He et al., 2016], por exemplo, reduz o passo α em marcos \mathcal{M} previamente definidos. Assim, a atualização da taxa de aprendizado em cada época t passa a ser dada por:

$$\alpha(t) = \begin{cases} \gamma\alpha(t-1) & \text{se } t \in \mathcal{M} \\ \alpha(t-1), & \text{se } t \notin \mathcal{M} \end{cases} \quad (3.6)$$

tal que $\gamma \in]0, 1[$ é um fator de redução e $\alpha(t)$ e $\alpha(t-1)$ são, respectivamente, as taxas de aprendizado na época atual e na época anterior.

Outro *scheduler* muito comum é o *cosine decay* [Loshchilov e Hutter, 2016; He et al., 2019], que propõe uma redução gradual da taxa de aprendizado segundo uma cossenóide:

$$\alpha(t) = \frac{\alpha(0)}{2} \left(1 + \cos \left(\frac{t\pi}{T} \right) \right) \quad (3.7)$$

onde $\alpha(0)$ é a taxa de aprendizado inicial e T é o número de épocas totais.

Analogamente, o *inverse exponential scheduler* [Jocher et al., 2020] propõe uma diminuição exponencial da taxa de aprendizado, em cada época t , de acordo com:

$$\alpha(t) = \alpha(0) \left(1 - 10^{\eta \left(1 - \frac{t}{T} \right)} \right) \quad (3.8)$$

onde η é uma constante de decaimento negativa.

Ainda que essas estratégias promovam vantagens no treinamento, um passo maior nas primeiras épocas também pode causar problemas em algumas circunstâncias. No que se refere ao aprendizado em grandes bases de dados, torna-se altamente desejável utilizar lotes com um número maior de exemplos, a fim de que seja possível paralelizar o treinamento em vários processos. Todavia, isso faz com que os gradientes oscilem ainda mais com uma taxa de aprendizado alta e propiciem uma estagnação das variáveis independentes em ótimos locais ruins [Goyal et al., 2017b; Vaswani et al., 2017].

Diante disso, recomenda-se a técnica *warm-up* [He et al., 2016], também conhe-

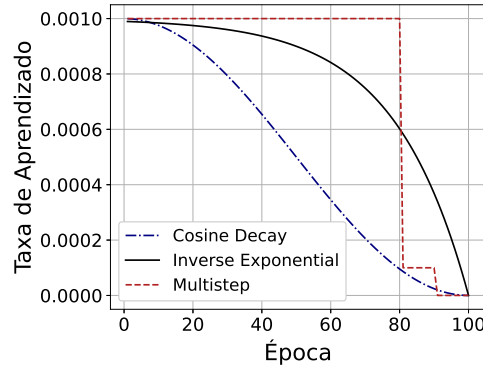


Figura 3.8: Comportamento dos diferentes tipos de *learning rate schedulers* para $\alpha(0) = 0,001$ e $T = 100$ épocas. No caso do *multistep*, tem-se um fator de redução $\gamma = 0,1$ e $\mathcal{M} = \{80, 90\}$, isto é, marcos em 80% e 90% das épocas totais. Quanto ao *inverse exponential*, assumiu-se uma constante de decaimento $\eta = -2$. O *cosine decay*, por sua vez, não contempla nenhum hiperparâmetro adicional.

cida como *burn-in*, que aumenta progressivamente a taxa de aprendizado nas primeiras τ épocas até um valor $\alpha(\tau)$, de modo a assegurar estabilidade na fase inicial do treinamento [Gotmare et al., 2018]. O crescimento da taxa de aprendizado em cada época também pode ser modelado por diversas funções, sendo a de maior relevância para o presente trabalho dada por:

$$\alpha(t) = \alpha(0) \left(\frac{t}{\tau} \right)^\kappa, \text{ se } t < \tau \quad (3.9)$$

onde κ é um expoente positivo [Redmon e Farhadi, 2017]. Para $t \geq \tau$, a taxa de aprendizado geralmente se comporta conforme o decaimento promovido por um *scheduler*, isto é, pela equação (3.6), (3.7) ou (3.8) (Figura 3.9).

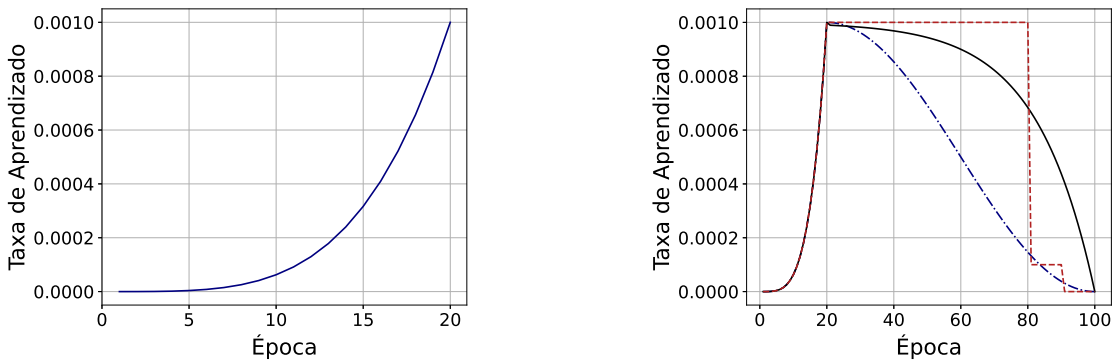


Figura 3.9: Técnica *warm-up* com $\kappa = 4$, $\tau = 20$ épocas, $T = 100$ épocas e $\alpha(0) = 0,001$ (à esquerda), bem como sua associação aos *learning rate schedulers* apresentados na Figura 3.8 (à direita).

3.2 Redes neurais artificiais

A rede neural artificial (RNA) é um algoritmo de aprendizado de máquina supervisionado inspirado, ainda que vagamente, no estudo científico do sistema nervoso e é comumente utilizada para tarefas de classificação. Mediante uma estrutura composta por unidades de processamento interligadas, esses modelos computacionais são capazes de aprender padrões complexos nos dados e de realizar extrapolações do conhecimento adquirido.

Trivialmente intituladas de neurônios artificiais, as unidades são dispostas em camadas e calculam funções matemáticas tipicamente não-lineares [Braga, 2000]. Cada neurônio estabelece uma conexão com os neurônios das camadas adjacentes e essas conexões, por sua vez, estão associadas a pesos sinápticos, que são parâmetros responsáveis por armazenar a experiência do modelo.

As arquiteturas das RNAs podem ser representadas visualmente por grafos ponderados, onde os vértices são os neurônios, as arestas são as conexões entre os neurônios e as ponderações relativas às arestas são os pesos. O fluxo das operações matemáticas da rede segue a orientação do grafo, que inicia em uma camada de entrada, passa por uma ou mais camadas intermediárias ou ocultas e encerra em uma camada de saída. Ambas as redes neurais da Figura 3.10, por exemplo, possuem quatro neurônios na camada de entrada, três na camada intermediária e um na camada de saída.

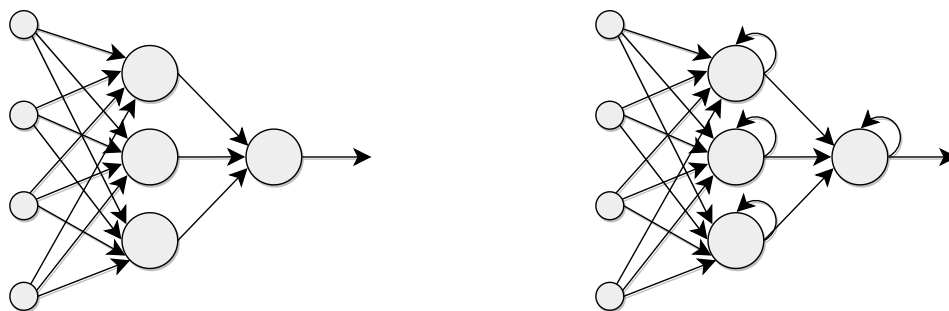


Figura 3.10: Se as conexões assumem um único sentido durante toda a arquitetura, essa rede é conhecida como rede de alimentação direta (*feedforward neural network*) (à esquerda). Caso haja ligações de realimentação, tem-se uma rede neural recorrente (*recurrent neural network*) (à direita).

3.2.1 Notação matemática

Uma rede neural com L camadas intermediárias retorna, para cada exemplo $\mathbf{x}^{(i)} \in \mathbb{R}^{n_x}$ do conjunto de dados de tamanho m , uma predição $\mathbf{y}^{(i)} \in \mathbb{R}^{n_y}$. Durante o processamento desses exemplos, são realizadas inúmeras operações de ponto flutuante. No que diz respeito à representação em grafos, cada seta que sai de um vértice indica

uma multiplicação do valor desse vértice pelo peso da aresta correspondente e cada seta que chega indica uma soma de várias ponderações.

Fundamentalmente, cada entrada de um neurônio j da camada l proveniente da conexão com o neurônio k da camada $l - 1$ é multiplicada por um peso sináptico $w_{jk}^{[l]}$. A soma ponderada dessas $n_h^{[l-1]}$ entradas é encaminhada para o núcleo do neurônio j , à qual é incorporada um viés $b_j^{[l]}$ (*bias*) e posteriormente modificada por uma função de ativação $f_j^{[l]}(\cdot)$. O resultado dessa transformação, denotado como ativação $a_j^{[l(i)]}$, é então propagado como uma das entradas para os neurônios da camada $l+1$ (Figura 3.11) e é dado por:

$$a_j^{[l(i)]} = f_j^{[l]}(z_j^{[l(i)]}) \quad (3.10)$$

tal que $z_j^{[l(i)]} = \sum_{k=1}^{n_h^{[l-1]}} a_k^{[l-1(i)]} w_{jk}^{[l]} + b_j^{[l]}$. Assim, as ativações da primeira camada podem ser expressadas como os valores das entradas propriamente ditas $\mathbf{a}^{[0(i)]} = \mathbf{x}^{(i)}$ e as ativações da última camada como os valores das saídas geradas $\mathbf{a}^{[L+1(i)]} = \hat{\mathbf{y}}^{(i)}$. Logo $n_x = n_h^{[0]}$ e $n_y = n_h^{[L+1]}$.

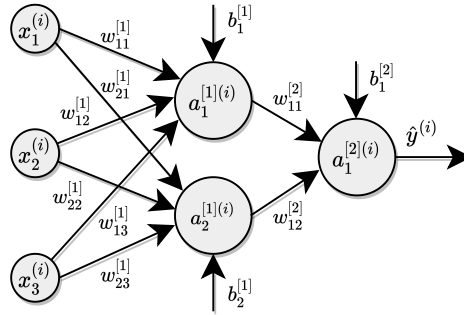


Figura 3.11: Representação clássica de uma rede neural com uma única camada intermediária ($L = 1$). Nesse caso, o grafo orientado ilustra o mapeamento de um exemplo $\mathbf{x}^{(i)} \in \mathbb{R}^3$ em uma saída $\hat{\mathbf{y}}^{(i)} \in \mathbb{R}$.

À medida que a dimensão da entrada e o número de neurônios crescem, o processo de computação da ativação de cada neurônio para cada exemplo a partir de (3.10), mesmo que implementado por laços de repetição, se torna bastante ineficiente. Sob essa concepção, adota-se uma representação matricial que permite realizar o processamento paralelo de múltiplos exemplos. Para fins de simplificação, as próximas equações e notações serão formuladas sobre o princípio do gradiente descendente clássico, isto é, considerando um único lote de tamanho m .

Seja $\mathbf{X} \in \mathbb{R}^{n_x \times m}$ a matriz de entrada, $\mathbf{A}^{[l-1]} \in \mathbb{R}^{n_h^{[l-1]} \times m}$ as ativações dos neurônios da camada $l - 1$ e $\mathbf{W}^{[l]} \in \mathbb{R}^{n_h^{[l]} \times n_h^{[l-1]}}$ e $\mathbf{b}^{[l]} \in \mathbb{R}^{n_h^{[l]}}$, respectivamente, os pesos e *bias* da camada l . Tem-se:

$$\mathbf{A}^{[l]} = \mathbf{f}^{[l]}(\mathbf{Z}^{[l]}) \quad (3.11)$$

onde $\mathbf{Z}^{[l]} = \mathbf{W}^{[l]}\mathbf{A}^{[l-1]} + \mathbf{b}^{[l]}$ e $\mathbf{f}^{[l]} \in \mathbb{R}^{n_h^{[l]}}$ é um conjunto de funções de ativação. Por fim, os rótulos são representados por $\mathbf{Y} \in \mathbb{R}^{n_y \times m}$ e as predições por $\hat{\mathbf{Y}} \in \mathbb{R}^{n_y \times m}$.

3.2.2 Redes neurais profundas

O conceito de aprendizado profundo, provém da concepção de redes neurais com múltiplas camadas intermediárias ($L \geq 2$), também conhecidas como *multilayer perceptrons* (MLPs) ou *deep feedforward networks* (Figura 3.12). Ao adicionar uma camada de neurônios na rede, tem-se uma expansão da arquitetura em profundidade, de modo que a entrada a ser propagada passe por uma nova etapa de processamento antes de gerar a saída.

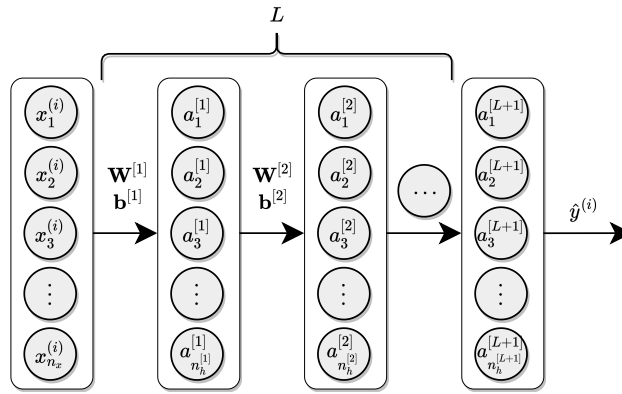


Figura 3.12: Representação simplificada de uma rede neural profunda com L camadas intermediárias. As conexões entre todos os neurônios foram omitidas para melhor visualização da arquitetura ao mapear $\mathbf{x}^{(i)} \in \mathbb{R}^{n_x}$ em $\hat{\mathbf{y}}^{(i)} \in \mathbb{R}^{n_y}$.

Uma rede neural suficientemente profunda, cujo aprendizado foi adquirido em um grande conjunto de dados rotulados, dispõe da capacidade necessária para representar funções de elevada complexidade. Assim, a maioria das tarefas que consistem em mapear uma entrada em uma saída e que estão ao alcance da execução humana com um certo nível de dificuldade, podem ser realizadas por meio de aprendizado profundo [Goodfellow et al., 2016].

Contudo, inúmeros desafios emergem ao se utilizar arquiteturas tão complexas. À medida que o número de camadas é ampliado, o número de parâmetros da rede neural conseqüentemente aumenta. Tal particularidade implica diretamente no custo computacional do modelo, uma vez que se tem um crescimento significativo no número de gradientes calculados durante o treinamento e no número de operações de ponto flutuante demandadas para a computação da saída da rede. Outros problemas serão discutidos nas próximas subseções, tal como potenciais soluções que atualmente tornam o aprendizado profundo a essência de diversas aplicações baseadas em inteligência artificial.

3.2.3 *Backpropagation*

Para mapear uma dada entrada \mathbf{X} em uma saída $\hat{\mathbf{Y}}$, a rede neural processa a informação recebida camada por camada. Essa fase é conhecida como propagação direta (*forward propagation*) e, durante o treinamento, fornece uma avaliação da função de custo, na qual retrata diretamente a qualidade dos parâmetros naquele dado momento. Tradicionalmente, esse custo é calculado de acordo com uma métrica de erro que compara a saída gerada $\hat{\mathbf{Y}}$ com a saída desejada \mathbf{Y} . Em problemas de regressão, por exemplo, é bastante comum utilizar a função de custo *mean squared error* (MSE):

$$\mathcal{J}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{m} \sum_{k=1}^m \left(y^{(i)} - \hat{y}^{(i)} \right)^2 \quad (3.12)$$

onde m é o número de exemplos contido na partição dos dados que se deseja avaliar o desempenho do modelo. Já em problemas de classificação, geralmente emprega-se a *categorical cross-entropy* (CCE):

$$\mathcal{J}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\frac{1}{m} \sum_{k=1}^m \sum_{j=1}^C y_j^{(i)} \log \left(\hat{y}_j^{(i)} \right) + \left(1 - y_j^{(i)} \right) \log \left(1 - \hat{y}_j^{(i)} \right) \quad (3.13)$$

onde C é o número de classes do problema. Para $C = 2$, tem-se o caso particular da *binary cross-entropy* (BCE).

Em virtude da necessidade de ajuste dos pesos e *bias* da rede, as derivadas da função de custo em relação a todos os parâmetros devem ser calculadas e fornecidas a algum método de otimização baseado em gradiente, como os discutidos na Subseção 3.1.5. Embora seja simples calcular analiticamente a expressão para o gradiente, avaliá-la numericamente pode ser computacionalmente inviável, principalmente se o número de parâmetros da rede neural for bastante elevado [Goodfellow et al., 2016].

Diante disso, Rumelhart et al. [1986], inspirados na regra delta de Widrow e Hoff [1960], propuseram o algoritmo supervisionado de retropropagação de erros (*backpropagation*) para um cálculo mais simples e eficiente do gradiente. Em suma, este método propõe uma fase de propagação reversa, denominada *backward propagation*, onde as informações provenientes do custo fazem o caminho inverso na rede para computar em cadeia as derivadas relativas a cada um dos parâmetros.

A derivada parcial da função de custo em relação a um determinado parâmetro permite mensurar a sensibilidade do desempenho da rede à pequenas variações desse parâmetro. Seja uma observação i , a componente do gradiente $\nabla \mathcal{L}$ referente ao peso

$w_{jk}^{[l]}$ é calculada com base na regra da cadeia de Leibniz (*chain rule*):

$$\frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial y^{(i)}} \frac{\partial y^{(i)}}{\partial a_j^{[L+1](i)}} \frac{\partial a_j^{[L+1](i)}}{\partial z_j^{[L+1](i)}} \frac{\partial z_j^{[L+1](i)}}{\partial a_j^{[L](i)}} \cdots \frac{\partial z_j^{[l+1](i)}}{\partial a_j^{[l](i)}} \frac{\partial a_j^{[l](i)}}{\partial z_j^{[l](i)}} \frac{\partial z_j^{[l](i)}}{\partial w_{jk}^{[l]}} \quad (3.14)$$

A expansão da derivada parcial evidencia o caráter recursivo do *backpropagation*, onde os termos derivativos contemplam informações de todas as camadas contidas no percurso da camada de saída $L + 1$ até a camada l referente ao peso $w_{jk}^{[l]}$. Alguns desses termos, no entanto, são redundantes e podem ser simplificados:

$$\frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\cancel{\partial y^{(i)}}} \frac{\cancel{\partial y^{(i)}}}{\cancel{\partial a_j^{[L+1](i)}}} \frac{\cancel{\partial a_j^{[L+1](i)}}}{\cancel{\partial z_j^{[L+1](i)}}} \frac{\cancel{\partial z_j^{[L+1](i)}}}{\cancel{\partial a_j^{[L](i)}}} \cdots \frac{\cancel{\partial z_j^{[l+1](i)}}}{\cancel{\partial a_j^{[l](i)}}} \frac{\cancel{\partial a_j^{[l](i)}}}{\partial z_j^{[l](i)}} \frac{\partial z_j^{[l](i)}}{\partial w_{jk}^{[l]}} \quad (3.15)$$

Finalmente, a derivada da função de custo para um dado peso sináptico $w_{jk}^{[l]}$ da rede neural pode ser expressa por:

$$\frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial z_j^{[l](i)}} \frac{\partial z_j^{[l](i)}}{\partial w_{jk}^{[l]}} \quad (3.16)$$

tal que $\frac{\partial z_j^{[l]}}{\partial w_{jk}^{[l]}} = a_k^{[l-1]}$. De maneira análoga, a derivada da função de custo para um dado viés $b_j^{[l]}$ pode ser definida por:

$$\frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial b_j^{[l]}} = \frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial z_j^{[l](i)}} \frac{\partial z_j^{[l](i)}}{\partial b_j^{[l]}} \quad (3.17)$$

tal que $\frac{\partial z_j^{[l]}}{\partial b_j^{[l]}} = 1$.

Sem perda de generalidade, (3.16) e (3.17) podem ser extendidas para um conjunto de observações a partir da média das derivadas da função de custo em cada um dos m exemplos:

$$\frac{\partial \mathcal{J}(\mathbf{Y}, \hat{\mathbf{Y}})}{\partial w_{jk}^{[l]}} = \frac{1}{m} \sum_{i=1}^m \frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial w_{jk}^{[l]}} \quad (3.18)$$

$$\frac{\partial \mathcal{J}(\mathbf{Y}, \hat{\mathbf{Y}})}{\partial b_j^{[l]}} = \frac{1}{m} \sum_{i=1}^m \frac{\partial \mathcal{L}(y^{(i)}, \hat{y}^{(i)})}{\partial b_j^{[l]}} \quad (3.19)$$

3.2.4 Funções de ativação

Funções de ativação são transformações não lineares utilizadas na entrada de cada neurônio que compõe a rede neural. Após a transformação, as saídas dos neurônios presentes na camada corrente são linearmente combinadas e posteriormente propagadas para as entradas dos neurônios da camada seguinte, permitindo ao modelo aprender tarefas mais complexas no final do treinamento.

A ausência da função de ativação restringe a rede neural a um modelo de regressão linear, uma vez que a sua saída será resultado de sucessivas transformações lineares da entrada. Contudo, a função de ativação não pode ser definida de modo arbitrário. Sua escolha também implica diretamente no aprendizado dos pesos e, portanto, deve ser realizada com cautela, contemplando aspectos da arquitetura da rede e propriedades particulares do problema a ser resolvido.

Ainda assim, algumas diretrizes são consideradas para tais escolhas. É muito comum, por exemplo, empregar uma mesma função de ativação para todos os neurônios da mesma camada ($f_j^{[l]}(\cdot) = f^{[l]}(\cdot) \forall j \in l$ -ésima camada), assim como utilizar funções diferentes entre as camadas intermediárias e final, visto que a última camada é responsável por modelar a saída segundo o número de classes do problema.

3.2.4.1 Identidade

A função de ativação identidade é a mais básica e existe apenas para sustentar a premissa de que cada neurônio tem como entrada uma soma ponderada à qual aplica uma função de ativação. De fato, utilizá-la significa apenas que não há ativação nos dados, ou seja, a entrada é exatamente igual a saída (Figura 3.13).

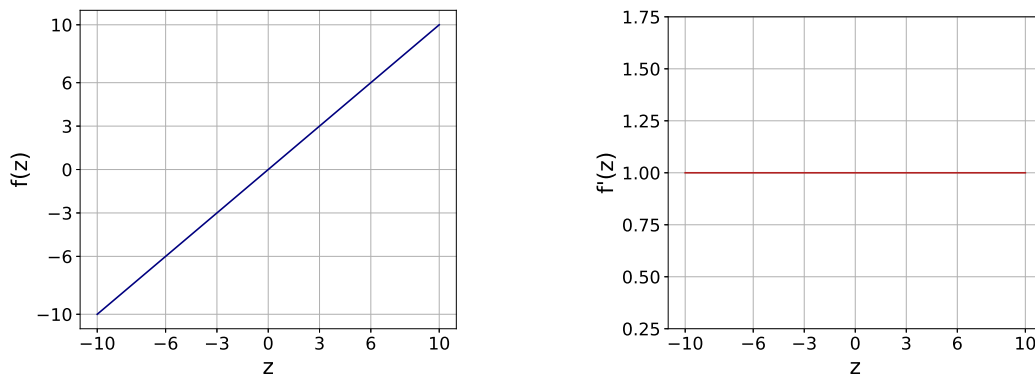


Figura 3.13: Função de ativação identidade (à esquerda) e sua derivada (à direita).

3.2.4.2 Sigmoide

Também conhecida como função logística, a função sigmoide é uma função suave e continuamente diferenciável bastante empregada nas redes neurais clássicas, especialmente na camada de saída daquelas projetadas para tarefas de classificação binária, visto que assume valores no intervalo $[0, 1]$ (Figura 3.14). Dado que ela é descrita por

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad (3.20)$$

uma propriedade bastante útil da função sigmoide é que sua derivada pode ser representada por meio de sua própria função:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \quad (3.21)$$

Todavia, sua utilização pode implicar em dois problemas principais. O primeiro deles é o problema de dissipação do gradiente (*the vanishing gradient problem*) que consiste no desaparecimento do gradiente durante o treinamento, dificultando o aprendizado da rede neural. Isso ocorre quando a entrada da função sigmoide é suficientemente grande em módulo, tornando a derivada bem próxima de zero (Figura 3.14). Quando a rede é composta por muitas camadas, a retropropagação do erro não ocorre por completo. Vários termos derivativos pequenos são multiplicados à medida que o erro se retropropaga para as camadas iniciais, diminuindo exponencialmente o gradiente e, conseqüentemente, não atualizando os pesos de forma adequada. Por fim, o segundo problema trata-se do mapeamento estritamente não-negativo, tal como a assimetria em relação à origem, o que nem sempre é desejável.

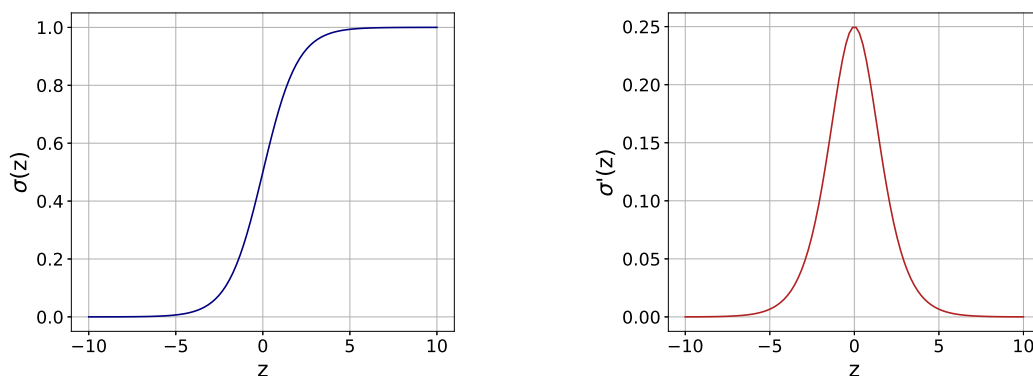


Figura 3.14: Função de ativação sigmoide (à esquerda) e sua derivada (à direita).

3.2.4.3 Softmax

A função softmax é uma função sigmoideal recomendada sempre que se deseja representar uma distribuição de probabilidades sobre uma variável discreta com um número previamente definido de valores ou categorias [Goodfellow et al., 2016]. Na prática, ela permite definir a probabilidade de uma entrada pertencer a uma classe em um problema de múltiplas classes, o que justifica bastante seu uso nos neurônios da camada de saída das redes. Matematicamente, a função exponencial é aplicada a cada elemento z_j do vetor de entrada $\mathbf{z} \in \mathbb{R}^C = [z_1, \dots, z_C]$ e normalizada pela soma de todas essas exponenciais:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{c=1}^C e^{z_c}} \quad \text{para } j = 1, \dots, C \quad (3.22)$$

onde C é o número de classes do problema de interesse.

3.2.4.4 Tangente hiperbólica

A tangente hiperbólica é uma função muito similar à sigmoide. Além de ambas serem continuamente diferenciáveis, elas ainda são intimamente relacionadas por:

$$\tanh(z) = 2\sigma(2z) - 1 \quad (3.23)$$

Diretamente, sua representação mais comum é:

$$\tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1} \quad (3.24)$$

e de sua derivada é:

$$\tanh'(z) = \frac{4}{(e^{-z} + e^z)^2} \quad (3.25)$$

Uma vez que a utilização da função sigmoide é viável, recomenda-se substituí-la pela tangente hiperbólica, que geralmente apresenta um melhor desempenho [Goodfellow et al., 2016]. Uma justificativa plausível para isso é a simetria da tangente hiperbólica em relação à origem, fruto do mapeamento de seu domínio para o intervalo $[-1, 1]$, que, por consequência, resolve o problema de não produzir ativações negativas (Figura 3.15). Em decorrência das derivadas nulas para valores extremos, o problema de dissipação do gradiente ainda ocorre em redes profundas.

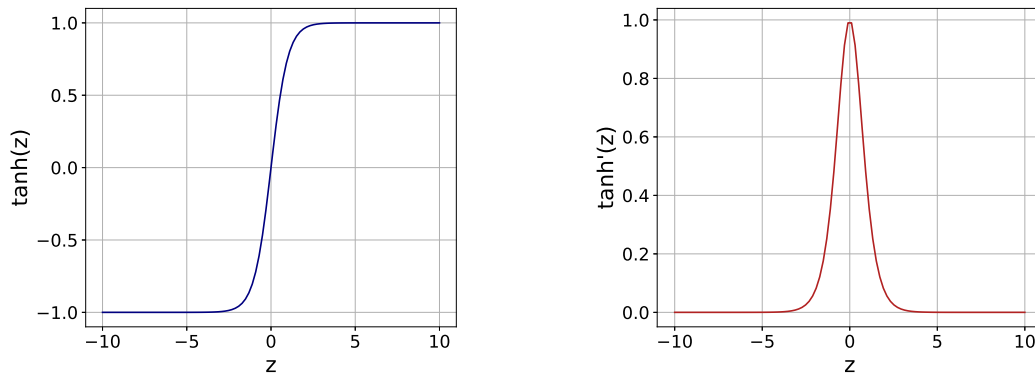


Figura 3.15: Função tangente hiperbólica (à esquerda) e sua derivada (à direita).

3.2.4.5 *Rectified linear unit* (ReLU)

A *rectified linear unit* (ReLU) é uma função de ativação por partes, cujo comportamento é linear para o domínio positivo e nulo para o domínio negativo (Figura 3.16). Tal composição abrange tanto propriedades desejáveis de uma função linear que facilitam o treinamento baseado em gradiente quanto características não lineares que permitem o aprendizado de relações mais complexas nos dados. Sua adoção, ainda que tardia por ser não diferenciável em zero, foi um grande marco na revolução das redes profundas, visto que ela é mais sensível à entrada, evita a saturação e, portanto, supera o problema de dissipação do gradiente, permitindo que os modelos de múltiplas camadas aprendam com maior velocidade e de forma efetiva [Nair e Hinton, 2010].

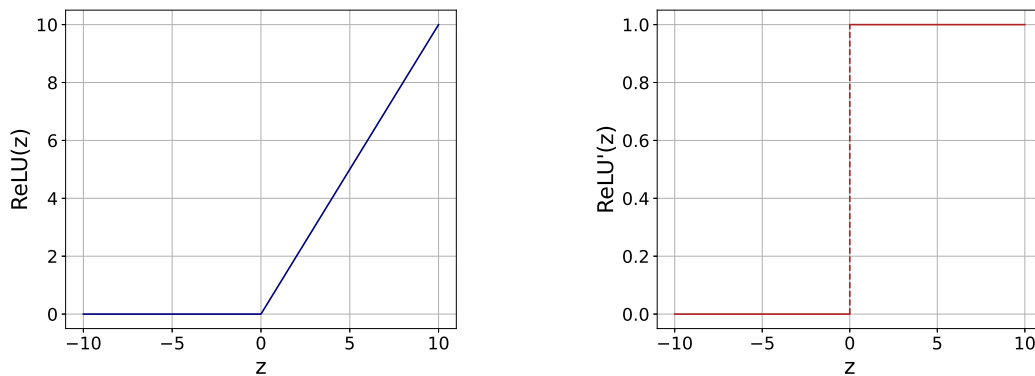


Figura 3.16: Função ReLU (à esquerda) e sua derivada (à direita).

Outra vantagem das unidades lineares retificadas é o seu menor custo computacional, uma vez que não é necessário calcular a função exponencial nas ativações:

$$\text{ReLU}(z) = \max(0, z) \quad (3.26)$$

e nem nas derivadas:

$$\text{ReLU}'(z) = \begin{cases} 0, & z < 0 \\ 1, & z > 0 \end{cases} \quad (3.27)$$

3.2.4.6 *Leaky rectified linear unit (LReLU)*

Um problema recorrente relacionado às unidades lineares retificadas em tarefas de visão computacional é o *dying ReLU problem*, que ocorre quando os neurônios da ReLU se tornam inativos e emitem zeros para qualquer entrada [Lu et al., 2019]. À medida que o número de neurônios inativos na rede neural aumenta, seu desempenho pode ser substancialmente afetado, posto que a derivada para o domínio negativo também é nula, impossibilitando a atualização dos pesos durante o treinamento. Uma maneira simples de resolver esse problema é garantir uma chance de recuperação a esses neurônios por meio de um pequeno gradiente positivo para as entradas negativas. A LReLU [Maas et al., 2013] propõe exatamente isso, onde a inclinação à esquerda de $z = 0$ passa a ser $\alpha = 0,01$, bem como seu gradiente para o mesmo domínio (Figura 3.17).

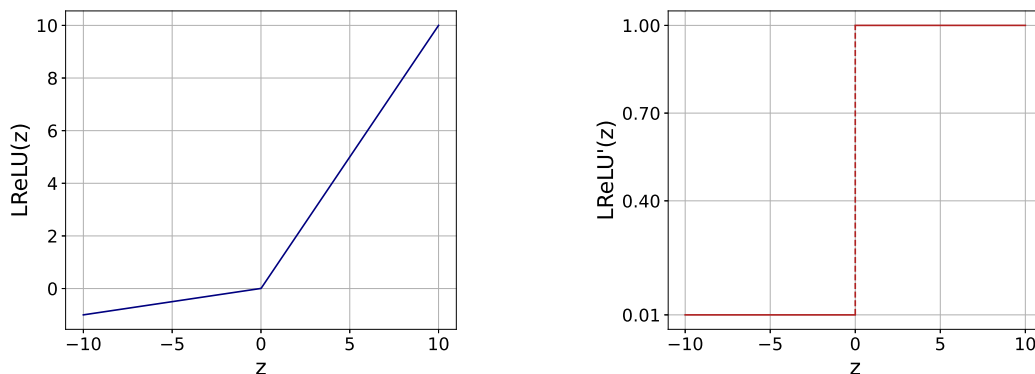


Figura 3.17: Função LReLU (à esquerda) e sua derivada (à direita).

3.2.4.7 *Exponential linear unit (ELU)*

Em busca de aperfeiçoar a ReLU em suas limitações, diversas outras variações foram propostas nos últimos anos. A *Exponential Linear Unit* (ELU), por exemplo, apresenta valores negativos capazes de garantir um estado de desativação robusto em termos de ruído, ou seja, saturam para um valor negativo e, assim, diminuem a variação e as informações propagadas [Clevert et al., 2016]. Para valores positivos, no entanto, a ELU preserva a eficácia no combate à dissipação do gradiente por meio da função identidade. No que se refere ao custo computacional, a ELU deixa a desejar devido a

incorporação em $z \leq 0$ de um termo exponencial na sua função:

$$\text{ELU}(z) = \begin{cases} z, & z > 0 \\ \alpha(e^z - 1), & z \leq 0 \end{cases} \quad (3.28)$$

e também na sua derivada:

$$\text{ELU}'(z) = \begin{cases} 1, & z > 0 \\ \alpha(e^z - 1) + \alpha, & z \leq 0 \end{cases} \quad (3.29)$$

onde, em geral, $\alpha = 1$. Os autores demonstram ainda que a escolha pela ELU promoveu uma convergência mais acelerada e também maiores precisões de classificação em comparação com as demais variações para as bases de dados CIFAR-10 e CIFAR-100.

3.2.4.8 *Rectified linear unit 6 (ReLU6)*

A ReLU6 [Krizhevsky e Hinton, 2010] é outra função de ativação inspirada na ReLU, cuja única diferença com a formulação clássica é a definição arbitrária do valor 6 como ativação máxima no domínio positivo (3.30), o que incentiva o modelo a aprender recursos anteriormente esparsos. Em virtude da sua baixa computação, ela é mais comum em redes projetadas para dispositivos móveis. A sua função é dada por:

$$\text{ReLU6}(z) = \min(\max(0, z), 6) \quad (3.30)$$

e sua derivada por:

$$\text{ReLU6}'(z) = \begin{cases} 0, & z < 0 \text{ ou } z > 6 \\ 1, & 0 < z < 6 \end{cases} \quad (3.31)$$

3.2.4.9 *Scaled exponential linear unit (SELU)*

Finalmente, uma das variações mais relevantes é a *Scaled Exponential Linear Unit* (SELU). Sua principal propriedade é induzir uma auto-normalização das ativações ao longo das camadas, mesmo sob a presença de ruídos e perturbações [Klambauer et al., 2017]. Enquanto normalizações tradicionais requerem um tratamento explícito, tais ativações convergem automaticamente para média zero e variância unitária, resultando em uma forte regularização, assim como aprendizados mais robustos, especialmente de

redes profundas. A SELU é expressa por:

$$\text{SELU}(z) = \begin{cases} \lambda z, & z > 0 \\ \lambda(\alpha e^z - \alpha), & z \leq 0 \end{cases} \quad (3.32)$$

e sua derivada por:

$$\text{SELU}'(z) = \begin{cases} \lambda, & z > 0 \\ \lambda\alpha e^z, & z \leq 0 \end{cases} \quad (3.33)$$

onde $\alpha = 1,6733$ e $\lambda = 1,0507$. A Figura 3.18 estabelece um paralelo entre as funções de ativação ELU, ReLU6 e SELU.

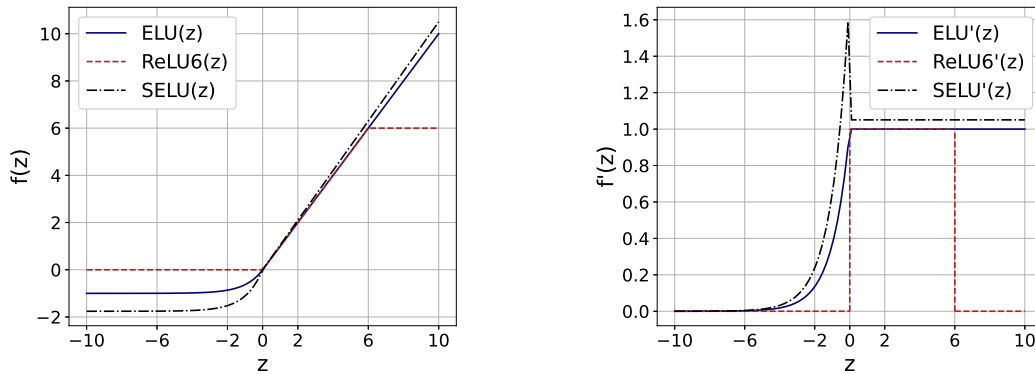


Figura 3.18: Variações da ReLU (à esquerda) e suas respectivas derivadas (à direita).

3.2.4.10 Softplus

Uma outra desvantagem da ReLU e de algumas de suas variações é que elas não podem aprender através de algoritmos baseados em gradiente nos exemplos para os quais suas ativações são exatamente zero [Goodfellow et al., 2016]. Dessa forma, diversas funções de ativação surgiram com o intuito de garantir a diferenciabilidade em todos os pontos do domínio, como é o caso da aproximação suave da ReLU denominada *Softplus* [Dugas et al., 2001]. Seu comportamento é descrito por:

$$\text{softplus}(z) = \ln(1 + e^z) \quad (3.34)$$

e sua propriedade de maior destaque é ter como derivada a função logística:

$$\text{softplus}'(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \quad (3.35)$$

A fim de evitar instabilidades numéricas ao produzir ativações quando $z \rightarrow \infty$ e ao produzir gradientes quando $z \rightarrow 0$, sugere-se uma expressão mais segura:

$$\text{softplus}(z) = \max(0, z) + \ln(1 + e^{-|z|}) \quad (3.36)$$

3.2.4.11 Swish

Tendo em vista que muitos especialistas têm favorecido a simplicidade e confiabilidade da ReLU, uma vez que as outras funções de ativação são inconsistentes em diferentes conjuntos de dados e modelos, membros do *Google Brain Team* propuseram a função *Swish* [Ramachandran et al., 2018], uma função de ativação suave e não-monotônica definida como:

$$\text{swish}(z) = z\sigma(z) \quad (3.37)$$

e de derivada de primeira ordem:

$$\text{swish}'(z) = z\sigma(z) + \sigma(z)(1 - z\sigma(z)) \quad (3.38)$$

Fundamentados pelo uso da função sigmoide na arquitetura da rede recorrente *Long Short-Term Memory* (LSTM) [Hochreiter e Schmidhuber, 1997], um mecanismo denominado *self-gating* é incorporado na concepção da função. A principal vantagem adquirida com essa técnica é que apenas uma entrada é necessária, enquanto que o *gating* tradicional exige vários escalares. Isso possibilita que a substituição da ReLU pela *Swish* seja muito mais simples e, portanto, de fácil aceitação pela comunidade.

Diferentemente da ReLU e da *Softplus*, essa função de ativação produz saídas negativas para pequenos valores de entrada negativos devido à sua monotonicidade, o que melhora significativamente o fluxo do gradiente [Ramachandran et al., 2018]. Essas comparações entre funções de ativação se estenderam para diversas arquiteturas em várias bases de dados de classificação de imagens e tradução por máquina e comprovaram que, além de atingirem desempenhos superiores na maioria dos casos, as ativações *Swish* viabilizam o aprendizado de redes ainda mais profundas.

3.2.4.12 Mish

Inspirada na *Swish*, Misra [2019] apresenta *Mish*, uma função auto-regularizada e não-monotônica e o atual estado da arte em funções de ativação. Matematicamente, ela pode ser representada como uma versão da *Softplus* com integração da técnica de *self-gating* proposta na *Swish*:

$$\text{mish}(z) = z \tanh(\text{softplus}(z)) \quad (3.39)$$

A ativação *Mish* é computacionalmente mais cara do que a ReLU e a *Swish*. O número de computações exponenciais na derivada (3.40) corrobora isso:

$$\text{mish}'(z) = \frac{e^z \omega}{\delta^2} \quad (3.40)$$

onde $\omega = 4(z + 1) + 4e^{2z} + e^{3z} + e^z(4z + 6)$ e $\delta = 2e^z + e^{2z} + 2$. Ciente disso, a autora também propõe uma versão otimizada da função, o que torna essa complexidade computacional irrisória na presença de GPUs.

A eficiência da *Mish* provém de uma composição de propriedades que ao longo do tempo foram evidenciadas como indispensáveis para o treinamento de redes neurais profundas. A mais clássica delas é ser ilimitada no domínio positivo para evitar tanto a saturação quanto a dissipação do gradiente. Além disso, a ligeira tolerância para valores negativos permite um melhor fluxo do gradiente e ainda preserva o limite rígido em zero proposto pela ReLU (Figura 3.19). Finalmente, a suavidade da curva em praticamente todo o domínio implica em uma alta capacidade de propagação de ativações ao longo das camadas, preservando a precisão e a generalização de forma mais efetiva.

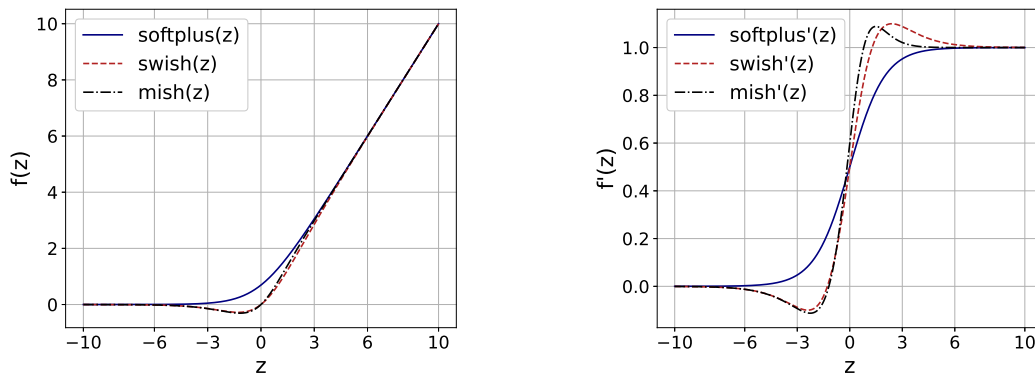


Figura 3.19: *Softplus*, *Swish* e *Mish* (à esquerda) e suas respectivas derivadas (à direita).

3.2.5 Inicialização de parâmetros

Algoritmos de otimização iterativos, tais como os baseados em gradiente, requerem a especificação de um ponto de partida para darem início ao ajuste dos parâmetros. Embora não pareça ser uma complicação a princípio, a escolha do procedimento de inicialização dos pesos e *bias* de uma rede neural interfere fortemente no seu treinamento. Um ponto inicial ruim é capaz de ocasionar problemas de instabilidade numérica e de convergência prematura em regiões de ótimos locais [Goodfellow et al., 2016].

A maioria das estratégias modernas de inicialização de parâmetros são heurísticas. Essa tendência é efeito da ausência de uma ampla compreensão acerca das influências

do ponto inicial na minimização do erro de teste. Em algumas circunstâncias, selecionar o ponto de partida que visa beneficiar o ajuste dos parâmetros no treinamento pode eventualmente prejudicar a generalização do modelo [Goodfellow et al., 2016].

Algumas propriedades, no entanto, devem ser consideradas. A primeira delas é que todos os parâmetros devem ser inicializados com valores distintos. Caso contrário, cada neurônio oculto receberá a mesma soma ponderada das entradas e, consequentemente, cada um aprenderá a identificar o mesmo padrão nos dados. Se porventura todos os valores forem zero, o que é ainda mais grave, todos os neurônios receberão sinais nulos para propagarem.

Para que essa quebra de simetria (*symmetry breaking*) seja possível, os pesos são comumente inicializados aleatoriamente segundo uma distribuição Gaussiana $\mathcal{N}(\mu, \sigma^2)$ ou uniforme $\mathcal{U}(a, b)$. E os vieses, por sua vez, bem como os demais parâmetros adicionais presentes em algumas arquiteturas específicas, são inicializados com valores constantes escolhidos heurísticamente [Goodfellow et al., 2016].

Outro fator que provoca bastante impacto na otimização dos parâmetros é o domínio de inicialização. Pesos muito grandes podem resultar em uma rede instável, onde grandes ativações são propagadas na fase de propagação direta e gradientes de grandes magnitudes são utilizados para atualizar os pesos na fase de retropropagação. No extremo, os pesos atingem valores tão elevados que computacionalmente tem-se um valor indefinido NaN (*not a number*). A esse fenômeno dá-se o nome de problema de explosão de gradientes (*exploding gradient problem*) [Hochreiter e Schmidhuber, 1997].

Nesse sentido, determinadas heurísticas foram propostas para definir o intervalo da distribuição. Glorot e Bengio [2010], por exemplo, sugerem uma inicialização normalizada em todas as matrizes de pesos $\mathbf{W}^{[l]}$ da rede neural:

$$\mathbf{W}^{[l]} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n^{[l-1]} + n^{[l]}}, \sqrt{\frac{6}{n^{[l-1]} + n^{[l]}}}\right) \quad (3.41)$$

onde $n^{[l-1]}$ e $n^{[l]}$ são, respectivamente, o número de entradas e saídas da camada l .

3.2.6 Transferência de aprendizado

No aprendizado profundo, o treinamento dos modelos com inicializações aleatórias (*training from scratch*) em grandes bases de dados pode ser bastante demorado. Milhões de parâmetros devem ser ajustados, assim como milhões de dados devem ser processados. Por ventura, esse não é o pior cenário. É possível ainda que não se tenha uma base de dados suficientemente grande para treinar o modelo complexo.

A transferência de aprendizado (*transfer learning*) é uma maneira de flexibilizar

a premissa de que os dados de treinamento e teste devem ser independentes e identicamente distribuídos, o que soluciona os dois problemas citados [Tan et al., 2018]. Essa técnica permite reutilizar características aprendidas para execução de uma tarefa em uma outra tarefa de natureza semelhante. Por exemplo, suponha que há um modelo treinado para identificar cachorros em imagens e que se deseja treinar um novo modelo para reconhecimento de leões. Uma possibilidade aqui é considerar o conhecimento adquirido anteriormente em cachorros e aproveitar em leões, posto que ambos são quadrúpedes, mamíferos e apresentam várias outras características em comum.

A metodologia da transferência de aprendizado pode ser decomposta nas seguintes etapas [Ribani e Marengoni, 2019]:

1. Seleção do modelo pré-treinado: diversas universidades e instituições de pesquisa disponibilizam modelos previamente treinados em grandes bases de dados populares. Assim, basta escolher um desses modelos que possui relação direta com a tarefa de interesse.
2. Reutilização dos parâmetros: o modelo pré-treinado deve ser copiado integralmente para outro modelo e todas ou algumas de suas camadas devem ser congeladas, de modo que alguns parâmetros não se atualizem (parâmetros não-treináveis) e que a experiência nos níveis de abstração desejados sejam preservadas.
3. Reestruturação do modelo: como a última camada modela a saída do problema, ela é geralmente substituída por uma camada com as propriedades do novo problema. Opcionalmente, pode-se ainda adicionar mais camadas após as camadas congeladas, sendo essas inicializadas de modo aleatório.
4. Adaptação para a nova tarefa: por fim, o modelo obtido deve ser refinado para o conjunto de dados da tarefa alvo. Esse ajuste fino (*fine-tuning*) normalmente é realizado através de um treinamento com um menor número de épocas e uma menor taxa de aprendizado. Antes dessa etapa, as camadas provenientes do modelo pré-treinado podem ser descongeladas para permitir a adaptação tanto das camadas adicionadas anteriormente quanto dos recursos transferidos.

3.2.7 Regularização

Do mesmo modo que os algoritmos de aprendizado de máquina tradicionais, as redes neurais sofrem frequentemente com o *overfitting* durante o processo de treinamento. Em arquiteturas profundas, esse problema é ainda mais usual, dado que a quantidade elevada de parâmetros está intrinsecamente relacionada a um modelo complexo. Complexidade, entretanto, pode ser sinônimo de superdimensionamento em alguns casos,

no qual o modelo tem capacidade suficiente para descrever qualquer observação do conjunto de treinamento mas sem capturar verdadeiramente o fenômeno pretendido.

Estratégias de regularização foram desenvolvidas justamente para contornar esse problema. Elas atuam sobre o compromisso viés e variância com a finalidade de reduzir o erro de generalização, mesmo que à custa de um aumento no erro de treinamento [Goodfellow et al., 2016]. Em concordância com a navalha de Occam [Blumer et al., 1987], que sugere a escolha da solução mais simples dentre duas ou mais equivalentes, algumas técnicas de regularização podem também reduzir consideravelmente a complexidade do modelo sem que haja depreciação substancial no seu desempenho.

3.2.7.1 Aumento de dados

A falta de generalização advém da incapacidade do modelo de priorizar, em um conjunto de dados limitado, somente as características que modelam o evento pretendido como um todo, isto é, sem particularidades exclusivas de algumas observações. A solução mais trivial para esse dilema surge espontaneamente na direção da aquisição de mais exemplos para treinamento. Um maior volume de dados implicaria em maior variabilidade dos padrões de interesse e, portanto, maiores chances do modelo, por mais complexo que seja, ajustar adequadamente seus parâmetros.

Na prática, isso nem sempre é possível. Em muitas ocasiões, o processo contínuo de coleta de dados, assim como sua rotulação no caso do aprendizado supervisionado, pode ser bastante custoso ou até completamente inviável. Com o propósito de viabilizar o aumento do número de exemplos de treinamento sem que seja preciso obter novos dados, normalmente emprega-se a técnica de expansão de dados, na qual realiza diversas operações nos dados originais para gerar exemplos artificialmente.

Em áreas como reconhecimento de objetos [Shorten e Khoshgoftaar, 2019] e reconhecimento de fala [Jaitly e Hinton, 2013], essa abordagem já foi validada inúmeras vezes, sendo inclusive identificada como uma componente essencial nos algoritmos estado da arte baseados em redes neurais profundas. No que tange a visão computacional, tema referente ao presente trabalho, são realizadas diversas transformações nas imagens, como rotação, translação, espelhamento, adição de ruído Gaussiano, redimensionamento e conversão entre espaços de cores.

3.2.7.2 Penalidades

No ponto de vista da otimização, regularizar corresponde a impor restrições sobre o espaço das variáveis de decisão, a fim de priorizar certas particularidades na solução durante a busca. Tipicamente, a resolução de um problema restrito deriva de sua

transformação em um problema irrestrito equivalente, tal que qualquer violação de alguma restrição é penalizada no valor da função objetivo [Batista, 2020].

De modo similar aos métodos de penalidades, regularizadores baseados em propriedades dos parâmetros também se fundamentam na incorporação de restrições ao problema de otimização. Nesse contexto, um objetivo adicional ψ é agregado à função de custo \mathcal{J} com o intuito de induzir a minimização da complexidade do modelo em conjunto com o erro de treinamento.

Em redes neurais, esse novo termo incide exclusivamente nos pesos sinápticos, visto que regularizar os *bias* pode fomentar intensamente a ocorrência de *underfitting* [Goodfellow et al., 2016]. A função de custo regularizada pode então ser expressa por:

$$\tilde{\mathcal{J}}(\mathbf{W}; \mathbf{Y}, \hat{\mathbf{Y}}) = \mathcal{J}(\mathbf{Y}, \hat{\mathbf{Y}}) + \lambda\psi(\mathbf{W}) \quad (3.42)$$

onde \mathbf{W} representa os pesos de todas as camadas da rede e $\lambda \in [0, \infty[$ um hiperparâmetro de regularização, que denota a proporção de contribuição da penalidade no valor total da função de custo. Valores de λ grandes e $\lambda = 0$ se referem, respectivamente, a uma alta regularização e à ausência de regularização durante o treinamento do modelo.

Na fase de propagação direta, uma rede neural com pesos de maior magnitude pondera intensamente as ativações e, por consequência, pode propagar qualquer evidência presente na entrada, inclusive ruídos locais. Em contrapartida, uma rede com pesos pequenos teve seu aprendizado construído apenas por padrões recorrentes nos dados e, portanto, é mais robusta às flutuações na entrada [Data Science Academy, 2019]. Diante disso, assumir a norma dos pesos como objetivo complementar é plenamente justificável.

A regularização ℓ_2 , também conhecida como *weight decay*, *ridge regression* e *Tikhonov regularization* [Tikhonov, 1963], por exemplo, atribui a esse termo uma soma escalonada dos quadrados de todos os pesos da rede neural:

$$\psi(\mathbf{W}) = \frac{1}{2m} \|\mathbf{W}\|_2^2 \quad (3.43)$$

sendo $\|\mathbf{W}\|_2^2 = \sum_{l=1}^{L+1} \sum_{j=1}^{n_h^l} \sum_{k=1}^{n_h^{l-1}} (w_{jk}^{[l]})^2$ e m o número de observações do conjunto de treinamento.

Outra alternativa para penalizar a magnitude dos parâmetros é a regularização ℓ_1 ou *least absolute shrinkage and selection operator* (LASSO) [Tibshirani, 1996], que considera para minimização a soma escalonada dos valores absolutos dos pesos:

$$\psi(\mathbf{W}) = \frac{1}{m} \|\mathbf{W}\|_1 \quad (3.44)$$

no qual $\|\mathbf{W}\|_1 = \sum_{l=1}^{L+1} \sum_{j=1}^{n_h^l} \sum_{k=1}^{n_h^{l-1}} |w_{jk}^{[l]}|$. Diferentemente da regularização ℓ_2 , essa abordagem resulta em soluções mais esparsas, isto é, redes com subconjuntos de pesos iguais a zero [Goodfellow et al., 2016].

3.2.7.3 Normalização em lote

A normalização em lote (*batch normalization*) foi proposta para acelerar o treinamento de redes neurais profundas. No entanto, o uso desta técnica também resultou em outros benefícios, como um grande potencial de regularização [Luo et al., 2018], robustez à configuração de hiperparâmetros e capacidade de evitar problemas de desaparecimento e explosão de gradientes [Santurkar et al., 2018]. Essencialmente, essa abordagem normaliza as entradas de cada camada oculta, de forma que haja determinada invariância das camadas à escala das ativações das camadas anteriores.

Dada uma rede profunda cujos pesos são otimizados pelo método do gradiente descendente em lote, tem-se a seguinte normalização da soma ponderada $\mathbf{z}^{[l]}$ para um exemplo i contido no lote $\mathbb{B} \in \mathbb{R}^{m'}$:

$$\mathbf{z}_n^{[l](i)} = \frac{\mathbf{z}^{[l](i)} - \mu_{\mathbb{B}}}{\sqrt{\sigma_{\mathbb{B}}^2 + \epsilon}} \quad (3.45)$$

onde ϵ é uma constante para assegurar estabilidade numérica, $\mu_{\mathbb{B}}$ é a média da soma ponderada no lote:

$$\mu_{\mathbb{B}} = \frac{1}{m'} \sum_{i=1}^{m'} \mathbf{z}^{[l](i)} \quad (3.46)$$

e $\sigma_{\mathbb{B}}^2$ é a variância da soma ponderada no lote:

$$\sigma_{\mathbb{B}}^2 = \frac{1}{m'} \sum_{i=1}^{m'} (\mathbf{z}^{[l](i)} - \mu_{\mathbb{B}})^2 \quad (3.47)$$

A fim de evitar que todas as camadas tenham distribuições de média 0 e desvio padrão 1, os valores normalizados são transformados linearmente por:

$$\tilde{\mathbf{z}}^{[l](i)} = \gamma \mathbf{z}_n^{[l](i)} + \beta \quad (3.48)$$

em que γ é um parâmetro de escala e β é um parâmetro de deslocamento, ambos ajustados durante o processo de treinamento.

3.3 Redes neurais convolucionais

Nos últimos anos, uma classe de redes neurais artificiais que vem sendo aplicada com bastante sucesso nas áreas de visão computacional e processamento de linguagem natural é a rede neural convolucional [LeCun et al., 1989b]. Inspirada no córtex visual dos animais, essa arquitetura é bastante adequada para processar dados de topologia em grade, como é o caso unidimensional das séries temporais e o caso bidimensional das imagens [Goodfellow et al., 2016].

No contexto de visão computacional, as redes convolucionais partem da premissa de que as entradas são imagens com largura, altura e profundidade, ou seja, tratam-se de redes especializadas, projetadas e ajustadas para trabalhar em tarefas relacionadas à visão humana, tipicamente classificação de objetos. Essa escalabilidade para imagens ou até mesmo para quadros de vídeos ocorre devido ao processo de separação da imagem em pixels de entrada, conforme apresentado na Figura 3.20.

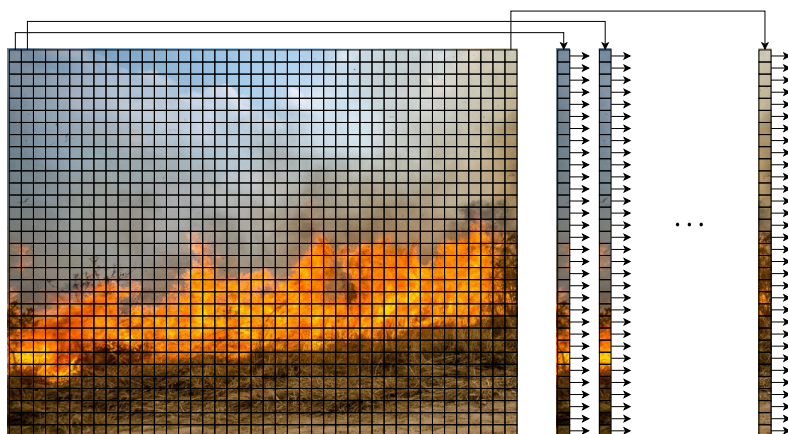


Figura 3.20: Transformação de uma imagem em w vetores de entrada com h pixels cada. Nesse caso, a imagem está no espaço de cores RGB e, portanto, cada pixel é uma tupla de 3 inteiros de 8 bits (R,G,B), sendo (0,0,0) preto e (255,255,255) branco.

Suponha que uma imagem RGB de dimensões 500×500 pixels seja entrada de uma rede neural clássica, o que é bastante modesto em comparação com a qualidade atual das fotos. Isso significa que o vetor de entrada teria 250.000 elementos em cada um dos canais de cores. Cada um desses elementos então seria conectado aos neurônios na primeira camada escondida. Se essa camada tivesse 1.000 neurônios, por exemplo, resultaria em 750 milhões de pesos a serem treinados apenas para essa parte da rede. Considerando que cada número de ponto flutuante ocupa tipicamente 8 bytes, esse grupo de pesos demandaria cerca de 6 bilhões de bytes para armazenamento, isto é, 6GB só para a primeira camada.

A solução encontrada pelas CNNs foi dispensar essa conectividade total por meio de uma estratégia de esparsidade. Ao invés de estabelecer uma conexão de cada neurônio com todos os outros neurônios da camada seguinte, cada neurônio é conectado apenas a um conjunto limitado de neurônios (campo receptivo) [Thom e Palm, 2013]. Além disso, outro fator considerado é a proximidade espacial dos pixels em uma imagem. Dois pixels que estão mais próximos um do outro são mais propensos a possuírem uma relação do que dois pixels mais distantes. Assim, como não há relação de cada pixel com cada neurônio, as redes convolucionais tornam o processamento de imagens computacionalmente gerenciável através de uma filtragem de conexões por proximidade (Figura 3.21).



Figura 3.21: A soma ponderada de entrada para um neurônio em uma rede neural totalmente conectada é afetada por todos os neurônios da camada anterior (à esquerda). Já em uma rede neural esparsa, como a rede convolucional, a entrada de um neurônio contém apenas influências de neurônios mais próximos (à direita).

A arquitetura de uma rede neural convolucional é composta por um elevado número de camadas ocultas dispostas em sequência. Juntamente com as camadas totalmente conectadas, ela normalmente dispõe de três outros tipos de camadas essenciais: (i) camada convolucional; (ii) camada de agrupamento e (iii) camada residual, cada qual com suas respectivas especificidades.

3.3.1 Camadas convolucionais

As camadas convolucionais são as camadas mais importantes da CNN e onde ocorre a maior parte do processamento computacional. Através de filtros lineares (*kernels*), os pixels da imagem são percorridos para produzir matrizes de saída conhecidas como mapas de características ou mapas de recursos (*feature maps*). No presente trabalho, ambos termos serão utilizados indiscriminadamente. Essa operação realizada pelo filtro é denominada convolução e é dada por:

$$M(i, j) = (\mathcal{I} * \mathcal{F})(i, j) = \sum_m \sum_n \mathcal{I}(m, n) \mathcal{F}(i - m, j - n) \quad (3.49)$$

em que \mathcal{I} é a imagem e \mathcal{F} é o filtro [Goodfellow et al., 2016].

A convolução é exemplificada a seguir, onde um filtro $\mathcal{F} \in \mathbb{R}^{2 \times 2}$ é aplicado a uma imagem $\mathcal{I} \in \mathbb{R}^{3 \times 3}$:

$$M = \mathcal{I} * \mathcal{F} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = \begin{bmatrix} 1 & 3 & 9 \\ 6 & 18 & 31 \\ 5 & 5 & 7 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \quad (3.50)$$

Todo valor m_{ij} do mapa de saída é uma soma ponderada dos pixels contidos em uma determinada vizinhança especificada pelo filtro, que se inicia no canto superior esquerdo da imagem:

$$\begin{bmatrix} 1 & 3 \\ 6 & 18 \end{bmatrix} \odot \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix} \quad (3.51)$$

$$m_{11} = 1 + 3 + 0 + 0 = 4$$

O cálculo do segundo elemento é efetuado após o filtro se deslocar um pixel para a direita na imagem:

$$\begin{bmatrix} 3 & 9 \\ 18 & 31 \end{bmatrix} \odot \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 9 \\ 0 & 0 \end{bmatrix} \quad (3.52)$$

$$m_{12} = 3 + 9 + 0 + 0 = 12$$

Uma vez que o limite horizontal da imagem foi atingido, o filtro retorna para o canto esquerdo da imagem e é deslocado um pixel para baixo:

$$\begin{bmatrix} 6 & 18 \\ 5 & 5 \end{bmatrix} \odot \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 6 & 18 \\ 0 & 0 \end{bmatrix} \quad (3.53)$$

$$m_{21} = 6 + 18 + 0 + 0 = 24$$

Por fim, o filtro se desloca um pixel para a direita novamente:

$$\begin{bmatrix} 18 & 31 \\ 5 & 7 \end{bmatrix} \odot \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 18 & 31 \\ 0 & 0 \end{bmatrix} \quad (3.54)$$

$$m_{22} = 18 + 31 + 0 + 0 = 49$$

Os mapas de recursos indicam regiões nas quais determinadas características da

entrada foram evidenciadas pelos filtros. Os valores reais dos filtros são alguns dos parâmetros da rede convolucional e, portanto, são otimizados durante o treinamento, de forma que o modelo aprenda a identificar e extrair características relevantes do conjunto de dados. Diversos filtros são utilizados em uma camada convolucional l , tal que cada mapa de recursos $M_j^{[l]}$ seja empilhado com os demais em profundidade, produzindo um conjunto de mapas $\mathbf{M}^{[l]}$. Posteriormente, esses mapas podem ainda ser transformados por funções de ativação.

3.3.1.1 *Padding*

Considere agora a seguinte convolução:

$$\begin{bmatrix} 1 & 3 & 3 & 9 & 0 \\ 7 & 9 & 4 & 7 & 4 \\ 4 & 3 & 5 & 5 & 9 \\ 6 & 6 & 0 & 4 & 0 \\ 1 & 7 & 2 & 5 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -3 & 4 & 1 \\ -7 & 0 & 8 \\ -10 & -4 & 3 \end{bmatrix} \quad (3.55)$$

De acordo com o mapa de recursos obtido, é possível perceber nitidamente que a saída tem uma dimensão menor que a entrada. Matematicamente, a saída tem tamanho $(h - k + 1) \times (w - k + 1)$, em que h e w são, respectivamente, a altura e a largura da entrada e k é a dimensão do filtro quadrado. Muitas vezes esse processo não é desejável, visto que leva à perda de informações, especialmente perto ou nas próprias bordas de um objeto.

Com o intuito de permitir que a saída continue sendo propagada para mais camadas convolucionais sem que haja diminuição das dimensões de entrada, um preenchimento (*padding*) é adicionado à matriz de entrada. A principal técnica, denominada *same padding*, consiste em adicionar $\frac{k-1}{2}$ zeros ao redor da entrada até que cada elemento da borda também possa estar no centro de uma convolução, o que sugere um valor de k ímpar. Aplicando esse princípio, tem-se:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 3 & 9 & 0 & 0 \\ 0 & 7 & 9 & 4 & 7 & 4 & 0 \\ 0 & 4 & 3 & 5 & 5 & 9 & 0 \\ 0 & 6 & 6 & 0 & 4 & 0 & 0 \\ 0 & 1 & 7 & 2 & 5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 15 & 1 & 10 & -6 & -25 \\ 24 & -3 & 4 & 1 & -28 \\ 21 & -7 & 0 & 8 & -21 \\ 22 & -10 & -4 & 3 & -18 \\ 20 & -4 & -6 & -2 & -14 \end{bmatrix} \quad (3.56)$$

3.3.1.2 *Stride*

O número de posições que o filtro vai deslocar horizontalmente e verticalmente é denominado passo (*stride*). Até então o passo estava definido como unitário. Considere agora o exemplo anterior com passo 2 e preenchimento:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 3 & 9 & 0 & 0 \\ 0 & 7 & 9 & 4 & 7 & 4 & 0 \\ 0 & 4 & 3 & 5 & 5 & 9 & 0 \\ 0 & 6 & 6 & 0 & 4 & 0 & 0 \\ 0 & 1 & 7 & 2 & 5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}_{\text{passo} = 2} = \begin{bmatrix} 15 & 10 & -25 \\ 21 & 0 & -21 \\ 20 & -6 & -14 \end{bmatrix} \quad (3.57)$$

Assim, a expressão atualizada para a dimensão do mapa de saída é:

$$\left(\frac{h + 2p - k}{s} + 1 \right) \times \left(\frac{w + 2p - k}{s} + 1 \right) \quad (3.58)$$

em que s é o tamanho do passo e p é o número de margens adicionadas como preenchimento na imagem de entrada.

3.3.2 Camadas de agrupamento

É muito comum a presença de uma camada de agrupamento após uma camada convolucional. Essa estratégia é utilizada para reduzir o tamanho das matrizes resultantes da convolução, preservando os dados relevantes das camadas anteriores. Além de diminuir o custo computacional, há uma redução significativa de parâmetros a serem aprendidos pela rede, contribuindo também para o controle do sobreajuste. Existem dois tipos principais de camada de agrupamento:

1. Agrupamento médio (*average pooling*): cada valor da matriz de saída é obtido de maneira similar à operação de convolução, porém ao invés de realizar uma soma ponderada, o filtro extrai o valor médio da região [LeCun et al., 1989a,b]. Essa operação é exemplificada a seguir com um filtro $\mathcal{F} \in \mathbb{R}^{2 \times 2}$ e passo 2:

$$f_{\text{médio}} \left(\begin{bmatrix} 1 & 3 & 3 & 9 \\ 7 & 9 & 4 & 7 \\ 6 & 6 & 0 & 4 \\ 1 & 7 & 2 & 5 \end{bmatrix} \right) = \begin{bmatrix} 5 & 5.75 \\ 5 & 2.75 \end{bmatrix} \quad (3.59)$$

2. Agrupamento máximo (*max pooling*): consiste em reduzir a dimensão das camadas pegando o valor máximo de cada região delimitada pelo filtro. Sua aplicação vem obtendo melhores resultados, uma vez que pixels vizinhos são altamente correlacionados [Ciresan et al., 2011; Krizhevsky et al., 2012; Szegedy et al., 2015]. O resultado dessa operação para o exemplo anterior é dado por:

$$f_{\max} \left(\begin{bmatrix} 1 & 3 & 3 & 9 \\ 7 & 9 & 4 & 7 \\ 6 & 6 & 0 & 4 \\ 1 & 7 & 2 & 5 \end{bmatrix} \right) = \begin{bmatrix} 9 & 9 \\ 7 & 5 \end{bmatrix} \quad (3.60)$$

3.3.3 Blocos residuais

A profundidade da rede é muito relevante nas arquiteturas das CNNs. Quanto maior o número de camadas convolucionais, maior o número de características extraídas da imagem em diferentes níveis de abstração e, possivelmente, melhor o desempenho do modelo [Urban et al., 2016]. Entretanto, redes mais profundas são mais difíceis de treinar, visto que com o aumento da profundidade, a precisão fica saturada (gradiente infinitamente pequeno) e depois se degrada rapidamente.

Os blocos residuais, provenientes das redes neurais residuais (ResNets), facilitam o treinamento nesse caso e permitem que as arquiteturas sejam mais profundas [He et al., 2016]. Cada bloco residual adiciona uma conexão no fluxo de propagação que permite saltar q camadas convolucionais (Figura 3.22). Tais conexões possibilitam que camadas mais profundas recebam mapas de recursos das camadas mais superficiais, inibindo eventuais dissipações do gradiente. Li et al. [2018] relatam ainda que conexões de salto (*skip connections*) simplificam indiretamente as funções de perda e, por isso, auxiliam na otimização.

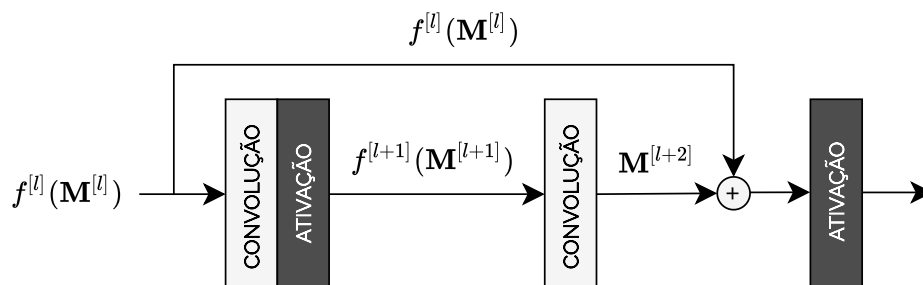


Figura 3.22: Seja um atalho de $q = 2$ após a camada l e que toda camada convolucional (cinza claro) é seguida por uma função de ativação $f^{[l]}$ (cinza escuro), a saída desse bloco residual será a transformação não-linear da soma do mapa de recursos da camada $l + 2$ com a ativação do mapa de recursos da camada l , isto é, $f^{[l+2]}(\mathbf{M}^{[l+2]} + f^{[l]}(\mathbf{M}^{[l]}))$.

3.3.4 Representação tridimensional

Um único filtro $\mathcal{F}_j^{[l]}$ não tem utilidade alguma no reconhecimento de objetos mais complexos. Assim, um conjunto de filtros $\mathcal{F}^{[l]}$ é aplicado nos mapas de saída de cada camada l da rede. Caso a camada l seja a primeira camada da arquitetura, $\mathcal{F}^{[l]}$ é aplicado na própria imagem de entrada. Sob essa perspectiva, uma representação visual comumente utilizada pela comunidade dispõe os filtros da mesma camada e os mapas de saída resultantes em volumes. Para isso, algumas considerações devem ser feitas:

1. Todos os $n^{[l+1]}$ filtros aplicados aos mapas de recursos $\mathbf{M}^{[l]}$ da camada l terão a mesma dimensão $\mathcal{F}_j^{[l]} \in \mathbb{R}^{n^{[l]} \times k^{[l]} \times k^{[l]}}$, bem como os mesmos hiperparâmetros (passo e preenchimento).
2. A profundidade $n^{[l+1]}$ do volume de saída $\mathbf{M}^{[l+1]}$ corresponde a cada um dos diferentes filtros aplicados à entrada. Neurônios na mesma linha ou na mesma coluna, mas em profundidades diferentes, representam o resultado da aplicação de filtros distintos à mesma parte da entrada (Figura 3.23).
3. Cada fatia $M_j^{[l]}$ da profundidade é um mapa de recursos e, portanto, o resultado da aplicação de um único filtro $\mathcal{F}_j^{[l]}$ à entrada.
4. Cada filtro irá abranger toda a profundidade da camada. Em outras palavras, se um filtro \mathcal{F} for aplicado a qualquer entrada $\mathcal{I} \in \mathbb{R}^{n \times h \times w}$ ou mapa $M \in \mathbb{R}^{n \times h \times w}$, o filtro terá dimensão $n \times k \times k$. Por outro lado, não importa a profundidade da camada anterior, a saída de um filtro sempre será dada pela equação (3.58).

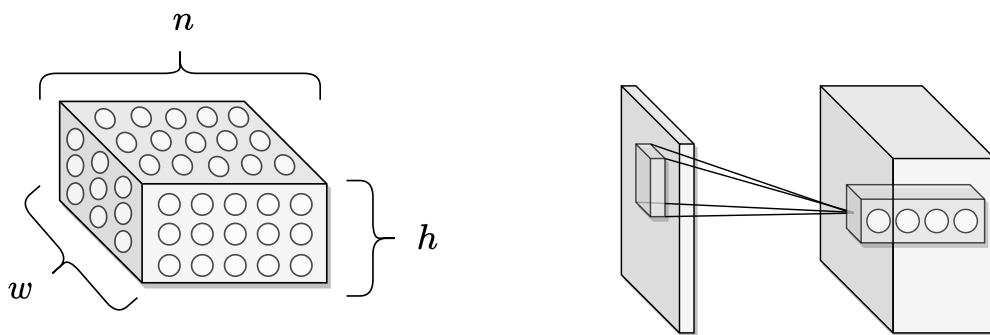


Figura 3.23: Representação de um conjunto de filtros convolucionais como volume (à esquerda). Saída de diferentes filtros aplicados à mesma parte da entrada (à direita). Em geral, os neurônios são omitidos dos volumes para uma melhor visualização da arquitetura da rede.

3.4 Detecção de objetos

O termo reconhecimento de objetos (*object recognition*) é amplamente utilizado para se referir a um conjunto de tarefas em visão computacional que estão relacionadas à identificação de objetos [Russakovsky et al., 2015]. Dentre essas tarefas pode-se destacar a classificação de imagens (*image classification*), localização de objetos (*object localization*), detecção de objetos (*object detection*), segmentação semântica (*semantic segmentation*) e segmentação de instância (*instance segmentation*).

A classificação de imagens permite determinar qual dos objetos de interesse está na cena, atribuindo um rótulo de classe para a imagem. A localização de objetos, por sua vez, é adicionada à classificação com o propósito de indicar não só a qual classe o objeto identificado pertence, mas também em que posição ele se encontra na imagem. Finalmente, a detecção de objetos estende o escopo da integração das duas tarefas anteriores ao tratar múltiplos objetos de classes distintas na mesma imagem.

Todas as tarefas descritas até o presente momento são categorizadas como problemas de predição esparsa. A esparsidade nesse contexto retrata a atribuição de poucos ou apenas um único rótulo à imagem. Em contraste, a segmentação semântica confere um rótulo para cada pixel da imagem, de modo que se obtenha uma região de pixels para cada classe de objetos presente na cena. Outra tarefa de predição densa é a segmentação de instância, que incorpora a distinção de diferentes objetos da mesma classe na atribuição de rótulos aos pixels. Esses conceitos podem ainda ser extrapolados para vídeos ao se considerar cada quadro (*frame*) como uma imagem independente.

Mesmo que a segmentação de instância disponha de uma formulação mais robusta, a detecção de objetos permite representar adequadamente problemas de monitoramento de eventos, uma vez que não é necessário obter os contornos das ocorrências para indicar a estimativa de suas respectivas localizações. Além disso, se porventura um algoritmo de aprendizado supervisionado for utilizado como solução para essas tarefas, os processos de rotulação das imagens para treinamento deverão conter tanto as classes dos objetos como as coordenadas das regiões que os compreendem. Isso de certa forma exigirá do projetista um esforço adicional para a segmentação de instância, onde cada objeto de cada imagem deverá ser demarcado por múltiplos pontos (polígono irregular) ao invés de apenas dois (retângulo).

3.4.1 Métricas de avaliação de desempenho

A avaliação de um modelo de aprendizado de máquina é essencial na obtenção da generalização desejável. Por esse motivo, a escolha de uma métrica quantitativa

que permita representar o real potencial do modelo torna-se imprescindível [Hossin e Sulaiman, 2015]. Em sintonia com a exposição feita na Subseção 3.1.1, de que cada tipo de tarefa requer uma métrica compatível, a detecção de objetos será examinada conforme uma perspectiva construtiva dos problemas de predição esparsos, isto é, a partir da classificação de imagens.

A habilidade de resolver tarefas de classificação e localização de múltiplos objetos concomitantemente concerne aos modelos de detecção a possibilidade de serem avaliados por métricas de ambas as tarefas. No que tange as métricas de classificação, a avaliação se restringiria apenas na capacidade do modelo de atribuir as classes corretas aos objetos. Por outro lado, as métricas de localização seriam encarregadas de mensurar a qualidade de ajuste das caixas delimitadoras independentemente das classes dos objetos. Em prol de uma avaliação mais pertinente, diversas adaptações dessas métricas foram propostas e deram origem a métricas quantitativas exclusivas para a tarefa de detecção de objetos [Everingham et al., 2010].

3.4.1.1 Interseção sobre união (IoU)

Interseção sobre união (IoU - *intersection over union*) é uma métrica de avaliação cujo objetivo é mensurar o quão boa é a previsão de qualquer algoritmo que forneça caixas delimitadoras preditas como saída [Everingham et al., 2010]. Para aplicá-la na análise de um algoritmo, é necessário identificar as seguintes caixas:

- Caixas delimitadoras reais: caixas delimitadoras rotuladas manualmente por projetistas antes do treinamento do modelo e que especificam onde está o objeto de interesse nas imagens do conjunto de dados.
- Caixas delimitadoras preditas: caixas delimitadoras identificadas pelo modelo na tentativa de localizar os objetos de interesse nas imagens.

A proximidade entre a caixa delimitadora predita e a caixa delimitadora real é calculada através de uma razão simples:

$$\text{IoU}(\text{real}, \text{predita}) = \frac{\text{área da interseção}}{\text{área da união}} \quad (3.61)$$

onde o numerador e o denominador representam a área da interseção e da união, respectivamente, entre a caixa delimitadora predita e a caixa delimitadora real.

Em um cenário real, é muito improvável que as coordenadas da caixa delimitadora predita correspondam exatamente às coordenadas da caixa delimitadora real. Devido a vários parâmetros dos modelos, uma correspondência completa e total entre

elas ($\text{IoU} = 100\%$) é irrealista. Assim, uma caixa delimitadora predita é considerada suficientemente confiável se o seu IoU for superior a um limiar t_{IoU} , cujo valor é tipicamente 50% [Hoiem et al., 2012]. Alguns algoritmos de detecção de objetos utilizam ainda esse limiar para suprimir todas as previsões de caixa com $\text{IoU} < t_{\text{IoU}}$, tal que um alto valor de t_{IoU} restringe as predições apenas a caixas delimitadoras de qualidade elevada.

3.4.1.2 Matriz de confusão

A matriz de confusão é uma representação popular de um conjunto de métricas para problemas de classificação, são elas verdadeiros positivos (vp), verdadeiros negativos (vn), falsos positivos (fp) e falsos negativos (fn). As colunas da matriz representam as classes reais e as linhas representam as classes previstas. Sendo assim, sua dimensão é $C \times C$, onde C é o número de classes do problema.

Suponha um problema de classificação binária, no qual se deseja verificar se há a ocorrência de um determinado evento ou não na imagem. A matriz de confusão correspondente é dada pela Tabela 3.1:

Tabela 3.1: Matriz de confusão com $C = 2$.

		Real	
		Sim	Não
Prevista	Sim	vp	fp
	Não	fn	vn

A diagonal principal da matriz corresponde aos acertos realizados pelo modelo. Nela, as imagens com o evento de interesse que foram classificadas corretamente pelo modelo são dadas como verdadeiros positivos e as imagens desprovidas do evento que foram classificadas corretamente pelo modelo são dadas como verdadeiros negativos. Em contrapartida, a diagonal secundária da matriz corresponde aos erros do modelo. Nela, as imagens com o evento que foram classificadas equivocadamente pelo modelo são dadas como falsos negativos e as imagens desprovidas do evento que foram classificadas equivocadamente são dadas como falsos positivos.

No caso particular da detecção de objetos, essas métricas assumem significados ligeiramente distintos. O número de verdadeiros positivos, por exemplo, é incrementado quando o modelo é capaz de encontrar o objeto corretamente. Um jeito de se verificar isso é comparando se o IoU das caixas predita e real é maior ou igual ao limiar t_{IoU} . O número de verdadeiros negativos, por sua vez, não são contabilizados. Se a proposta da tarefa é detectar objetos de interesse, deixar de detectá-los em situações que eles não existem não é mérito algum do modelo [Aidouni, 2019]. Um falso negativo ocorre

quando o modelo não é capaz de detectar um objeto que está presente na cena. Já o falso positivo é quando o modelo encontra um objeto que não está na imagem ou quando o IoU é inferior ao limiar t_{IoU} .

Embora cada métrica seja relevante para uma análise robusta, gerenciar três métricas na comparação de centenas de modelos pode ser bastante exaustivo. Em função disso, algumas outras métricas de avaliação quantitativas foram elaboradas a partir das métricas disponíveis na matriz de confusão [Hossin e Sulaiman, 2015]:

1. Precisão (*precision*): é a proporção de objetos detectados corretamente dentre todos os objetos detectados pelo modelo, conforme:

$$\text{precisão} = \frac{vp}{vp + fp} \quad (3.62)$$

2. Revocação (*recall*): é a proporção de objetos detectados corretamente dentre todos os objetos de interesse na cena, sendo definida como:

$$\text{revocação} = \frac{vp}{vp + fn} \quad (3.63)$$

3. Medida F_1 (F_1 -score): trata-se da média harmônica da precisão e da revocação e permite agregar todas as métricas anteriores em um único valor interpretável:

$$F_1 = 2 \cdot \frac{\text{precisão} \cdot \text{revocação}}{\text{precisão} + \text{revocação}} \quad (3.64)$$

É interessante ressaltar que uma média simples não é recomendada nesse contexto, visto que não penaliza valores extremos. Por exemplo, um modelo com precisão 0 e revocação 1 teria média 0,5. Já o F_1 para esse mesmo caso é 0.

3.4.1.3 Mean average precision (mAP)

O mapeamento explícito das probabilidades previstas por um classificador para o rótulo de classe propriamente dito é bastante capcioso. Em geral, essa modelagem é controlada por um limite de confiança. No caso do problema de classificação binária, por exemplo, essa atribuição de rótulos pode ser dada por uma função de Heaviside com descontinuidade em $\frac{1}{2}$, tal que todos os valores maiores ou iguais a $\frac{1}{2}$ são mapeados para uma classe e todos os demais são mapeados para outra [Rosenblatt, 1958].

Em circunstâncias atípicas, como o desbalanceamento de classes, em que o conjunto de dados apresenta uma classe com uma quantidade de exemplos significativamente maior do que as outras, possivelmente as C classes não poderão ser mapeadas

uniformemente no intervalo, isto é, com probabilidades iguais a $\frac{1}{C}$ [Fernández et al., 2018]. Ignorar esse fenômeno poderia eventualmente provocar uma interpretação não ótima das probabilidades previstas e, assim, prejudicar o desempenho do modelo.

Uma técnica bastante simples para determinar um limite de confiança apropriado para o problema é fazer uma análise da curva de precisão e revocação (*precision-recall curve*) [Buckland e Gey, 1994]. Essa curva retrata a relação de compromisso entre a precisão e a revocação para todos os valores de limite t_{conf} entre 0 e 1, onde o eixo das ordenadas é a precisão e o eixo das abscissas é a revocação. No cenário realista, essa curva exibe um padrão de zigzague. À medida que o número de falsos negativos diminui, a revocação aumenta. Contudo, a precisão cai com o aumento de falsos positivos e sobe com o aumento de verdadeiros positivos, o que caracteriza tal comportamento (Figura 3.24).

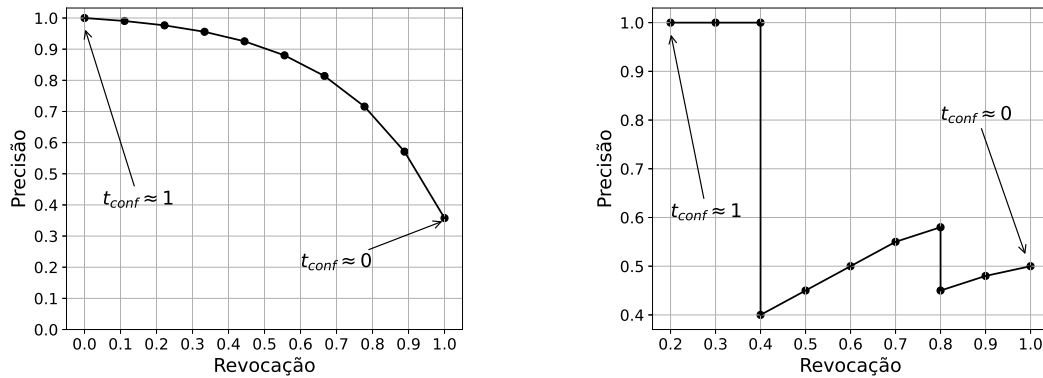


Figura 3.24: Exemplo de curva de precisão e revocação hipotética (à esquerda) e realista (à direita) [Hui, 2018]. Repare que, em ambos os casos, o modelo não produz falsos positivos no limite superior (precisão = 1) e não produz falsos negativos no limite inferior (revocação = 1).

A precisão média (AP - *average precision*), permite sumarizar toda informação disponível na curva de precisão e revocação em um único número para cada classe. Fundamentalmente, esse valor é dado pela média das avaliações de precisão em todos os valores de revocação, sendo tipicamente calculado pela interpolação da precisão com 11 pontos (0 a 1 com passo de tamanho 0,1), 101 pontos (0 a 1 com passo de tamanho 0,01) e com todos os pontos. A interpolação com todos os pontos, no entanto, é mais representativa e já foi priorizada em desafios renomados de visão computacional, como o *Pascal Visual Object Classes Challenge* (Pascal VOC) [Everingham et al., 2010] e o ILSVRC [Russakovsky et al., 2015].

Posto que todos os pontos de revocação são considerados, pode-se estimar a precisão média através da área sob a curva (AUC). De maneira análoga à regra do trapézio

composto utilizada para calcular numericamente uma integral, essa área pode ser definida como a soma de todas as áreas separadas sob a curva (Figura 3.25).

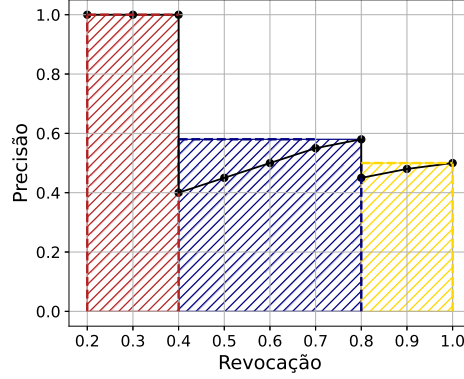


Figura 3.25: Na curva realista da Figura 3.24, a precisão média é dada pela soma de três áreas hachuradas. Cada área é delimitada após uma queda de precisão em um determinado ponto de revocação.

Matematicamente, tem-se:

$$AP = \sum_{r=0}^1 (r_{n+1} - r_n) \cdot \max(\text{precisão}(\tilde{r})) \quad (3.65)$$

tal que a diferença entre dois pontos de queda de revocação adjacentes ($r_{n+1} - r_n$) representa a base do retângulo e o valor máximo da precisão no intervalo $r_n \leq \tilde{r} \leq r_{n+1}$ é a altura do retângulo.

Finalmente, a média da precisão média em todas as C classes de objetos, denominada *mean average precision* (mAP) é dada por:

$$\text{mAP} = \frac{1}{C} \sum_{i=1}^C AP_i \quad (3.66)$$

O mAP está altamente correlacionado ao t_{IoU} . Atribuir um maior valor a esse limiar consiste em empregar maior rigor na aceitação de detecções como verdadeiros positivos. Esse número, por sua vez, incide nos cálculos de precisão e revocação, dos quais se deriva o AP e, por consequência, o mAP. Dessa forma, uma notação comum para expressar que o mAP foi calculado conforme um determinado valor de limiar t_{IoU} é $\text{mAP}@t_{\text{IoU}}$. Enquanto o desafio Pascal VOC considera um $\text{mAP}@0,50$, desafios que utilizam a base de dados COCO [Lin et al., 2014] calculam a média de mAP em diferentes limites IoU definidos entre 0,5 e 0,95 com um passo de 0,05, sendo tal métrica denotada como $\text{mAP}@0,5:0,95$ [Aidouni, 2019].

3.4.2 Detectores baseados em aprendizado profundo

Diante do notável progresso obtido pelas redes neurais convolucionais na classificação de imagens [Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016], uma tendência notável foi a incorporação dessas arquiteturas profundas em detectores de objetos [Girshick, 2015; Ren et al., 2016; Redmon et al., 2016]. A proposta de integração inicial considerava uma janela de pixels de tamanho fixo que percorria a imagem e, a cada nova posição, classificava a região delimitada pela janela com a rede convolucional treinada. Ao término desse processo, apenas as janelas cujas previsões apresentavam uma probabilidade de classe acima de um limiar previamente definido eram mantidas como detecções finais.

A principal limitação dessa abordagem ingênua é o tempo de processamento elevado. Executar o classificador inúmeras vezes pode inviabilizar qualquer aplicação de detecção em tempo real. Tal cenário motivou o surgimento de duas categorias de métodos [Li et al., 2020]:

1. Detectores de objetos de dois estágios: essa família de métodos tem como propósito dividir a detecção em duas etapas sequenciais. A primeira etapa consiste em empregar um algoritmo de proposta de região para identificar locais na imagem que têm grandes chances de conter os objetos desejáveis. Na etapa seguinte, cada uma dessas regiões são entregues a uma CNN, que reportará a existência ou não do objeto. Diferentemente do método da janela deslizante, esses detectores executam o classificador um número substancialmente menor de vezes.
2. Detectores de objetos de estágio único: também conhecidos como detectores de disparo único (*single-shot detectors*), esses métodos realizam tanto a classificação quanto a localização dos objetos em apenas uma etapa de *forward* da rede. A partir de predições em um conjunto padrão de caixas delimitadoras, torna-se dispensável o uso de algoritmos de proposta de região, que são tipicamente um gargalo computacional nos detectores de dois estágios [Jiao et al., 2019].

Não é possível ainda indicar uma classe de detectores mais promissora. Os detectores de estágio único apresentam arquiteturas mais simples e, conseqüentemente, processam um maior número de quadros por segundo (FPS - *frames per second*). Já os detectores de dois estágios mais modernos apresentam melhores desempenhos, especialmente em objetos pequenos [Lu et al., 2020]. No contexto de vigilância em larga escala, no entanto, há uma tendência por detectores mais rápidos e que requerem menos recurso computacional, já que possivelmente serão implementados em várias câmeras com processamento local nas regiões de interesse. Ademais, uma detecção tardia

pode implicar em danos humanos, ambientais e econômicos, o que sugere a escolha de detectores de objetos de um único estágio nesse caso.

3.4.3 *You Only Look Once (YOLO)*

Entre os algoritmos de detecção de objetos de estágio único mais renomados da literatura atualmente está o *You Only Look Once* (YOLO) [Redmon et al., 2016]. Como o próprio nome já sugere, uma única rede neural convolucional prevê simultaneamente caixas delimitadoras e probabilidades de classe diretamente da imagem em uma única execução. Dessa forma, a tarefa de detecção de objetos é tratada com uma visão holística, o que significa que cada detecção leva em consideração o ambiente e os outros objetos dispostos na cena.

Em sua primeira versão, YOLO redimensiona a imagem de entrada para 448×448 pixels e a fragmenta em uma grade de células de dimensão $S \times S$ (Figura 3.26). Cada uma das S^2 células prevê B caixas delimitadoras de objetos, juntamente com pontuações de confiança para cada caixa e C probabilidades de classe. Cada célula é responsável por detectar a existência de apenas um objeto. Se o centro desse objeto cair dentro da célula, essa célula é responsável por detectá-lo. Matematicamente, cada caixa delimitadora j de uma célula i é uma variável de otimização de dimensão \mathbb{R}^5 representada por:

$$\mathcal{B}_{ij} = [s_{ij}, x_{ij}, y_{ij}, h_{ij}, w_{ij}] \quad (3.67)$$

onde $s_{ij} = p_{ij} \text{IoU}_{ij}$ é uma pontuação de confiança do modelo, p_{ij} a probabilidade da caixa j conter um objeto, x_{ij} e y_{ij} são as coordenadas do centro da caixa j e h_{ij} e w_{ij} são, respectivamente, a altura e a largura da caixa j . Assim, cada célula i é definida como:

$$\mathcal{C}_i = [c_{i1}, c_{i2}, \dots, c_{ik}, \dots, c_{iC}, \mathcal{B}_{i1}, \mathcal{B}_{i2}, \dots, \mathcal{B}_{ij}, \dots, \mathcal{B}_{iB}] \quad (3.68)$$

onde c_{ik} é a probabilidade do objeto dentro da célula i ser da classe k , caso ele exista.

Se mais de um valor de confiança estiver acima de um certo limiar t_{conf} , a caixa delimitadora será definida conforme o maior valor de confiança s_{ij} , bem como a classe será definida conforme o maior valor de probabilidade de classe c_{ik} [Redmon et al., 2016]. Todavia, se o número de células S^2 for suficientemente grande, as caixas delimitadoras de células diferentes estarão muito próximas umas das outras e poderão ser atribuídas ao mesmo objeto.

De modo a evitar esse cenário, o YOLO emprega uma técnica conhecida como supressão não-máxima (NMS - *non-maximum suppression*) após a saída da rede. Essa operação seleciona a caixa com maior probabilidade de detecção p_{ij} e elimina as caixas

atribuídas ao mesmo objeto que mais a sobrepõem, isto é, as caixas cujo IoU com a caixa de referência excede t_{IoU} (Figura 3.27). Desconsiderando a caixa selecionada anteriormente, o processo se repete até que todas as caixas restantes sejam processadas.



Figura 3.26: Grade de células $S \times S$ com $S = 13$. Cada uma das células é colorida conforme a classe que tem a maior probabilidade prevista nessa célula, gerando um mapa de probabilidades de classe (à direita). Nesse exemplo, células com alta probabilidade de conter fumaça são coloridas de vermelho.



Figura 3.27: Apenas as caixas que o modelo atribuiu uma confiança acima de t_{conf} são mantidas, onde as de maior espessura indicam maior confiança (à esquerda). Como ainda são muitas caixas para um mesmo objeto, a melhor é selecionada através da supressão não-máxima (à direita).

3.4.3.1 Caixas âncora

Em comparação com os detectores baseados em região da época [Girshick, 2015; Ren et al., 2016], o YOLO ainda apresentava um maior erro de localização. Esforços nessa direção originaram o YOLOv2, uma segunda versão do YOLO, mais rápida e mais robusta [Redmon e Farhadi, 2017]. Mediante uma série de aperfeiçoamentos, o YOLOv2 se tornou o estado da arte em sistemas de detecção de objetos em tempo real por um período considerável.

Uma das principais propostas responsáveis pela eminente melhoria de desempenho foi incorporar caixas âncora (*anchor boxes*) na arquitetura da rede. Essas estruturas são basicamente caixas com proporção e razão de aspecto pré-definidos, no qual suas localizações estão vinculadas à posição central de cada célula da grade. O uso de várias caixas âncora expande a dimensão da variável de otimização presente na equação (3.67) e, por consequência, permite que várias classes sejam associadas a cada célula. Em outras palavras, torna-se possível detectar objetos de classes distintas mesmo quando estes se encontram sobrepostos.

A fim de construir boas caixas âncora iniciais, o algoritmo de agrupamento K-médias é empregado para percorrer o conjunto de dados e identificar os locais com maior probabilidade de conter objetos [Redmon e Farhadi, 2017]. Dado que a distância Euclidiana como função de similaridade pode gerar eventos indesejáveis, como erros proporcionais às dimensões da caixa, os autores propuseram uma métrica própria:

$$d(\text{caixa}, \text{centroide}) = 1 - \text{IoU}(\text{caixa}, \text{centroide}) \quad (3.69)$$

O número de caixas âncora, que coincide com o número de grupos (*clusters*), foi definido em $K = 5$ pelo Método do Cotovelo (*Elbow Method*), uma heurística que se baseia no compromisso entre complexidade e desempenho. Em uma versão posterior (YOLOv3), foram geradas analogamente 4 caixas âncora adicionais [Redmon e Farhadi, 2018].

3.4.3.2 Treinamento multi-escala

Outra novidade adicionada a partir da segunda versão do YOLO foi o treinamento multi-escala (*multi-scale training*) [Redmon e Farhadi, 2017]. Em oposição ao YOLOv1, cuja entrada é fixa em 448×448 pixels, a resolução de entrada passa a ser alterada no decorrer do treinamento. A cada 10 iterações do gradiente descendente em lote, a rede escolhe aleatoriamente um novo tamanho de dimensão para a imagem, a redimensiona e o treinamento continua.

Esse regime de múltiplas escalas induz a rede a aprender informações úteis acerca das classes em uma variedade de dimensões de entrada. O mesmo objeto em diferentes escalas implicará, portanto, em resultados fundamentalmente diferentes, já que as caixas delimitadoras irão se expandir ou diminuir. Como a rede reamostra a imagem por um fator de 32, a dimensão de entrada pode ser parametrizada como $32\rho \times 32\rho$, onde a menor resolução possível é 320×320 ($\rho = 10$) e a maior é 608×608 ($\rho = 19$).

3.4.3.3 Função de perda

De acordo com a discussão apresentada na Subseção 3.1.5, algoritmos de aprendizado de máquina minimizam uma função de perda para ajustar seus parâmetros aos

dados de treinamento. A função de perda do YOLOv3 é o resultado da soma de três componentes [Mostofa et al., 2020]:

1. Perda de localização: mede os erros nas posições e tamanhos das caixas delimitadoras previstas através da soma dos quadrados dos resíduos. No entanto, como não é interessante ponderar erros absolutos iguais em caixas de tamanhos distintos, o YOLO prevê a raiz quadrada da largura e da altura [Redmon et al., 2016]. Além disso, apenas a caixa responsável por detectar o objeto é considerada e a perda é dada por:

$$\begin{aligned} \mathcal{L}_{\text{loc}} &= \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \\ &\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \end{aligned} \quad (3.70)$$

onde λ_{coord} é responsável por intensificar a perda nas coordenadas da caixa e I_{ij}^{obj} é um indicador de caixa definido por:

$$I_{ij}^{\text{obj}} = \begin{cases} 1, & \text{se a } j\text{-ésima caixa na célula } i \text{ é responsável por detectar o objeto} \\ 0, & \text{caso contrário} \end{cases} \quad (3.71)$$

É interessante evidenciar que a notação assumida considera as variáveis com acento circunflexo como valores previstos pelo modelo e as variáveis sem acento circunflexo como valores de referência (rótulos).

2. Perda de confiança: permite mensurar a confiança do modelo ao determinar que uma caixa contém um objeto e o quão precisa ele acredita que essa caixa prevista está. Se um objeto for detectado na caixa, a perda de confiança será:

$$\mathcal{L}_{\text{conf}} = \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \mathcal{L}_{\text{BCE}}(s_{ij}, \hat{s}_{ij}) \quad (3.72)$$

tal que \mathcal{L}_{BCE} é o caso binário da função de entropia cruzada (3.13). Caso o objeto não seja detectado na caixa, a perda de confiança é:

$$\mathcal{L}_{\text{conf}} = \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{noobj}} \mathcal{L}_{\text{BCE}}(s_{ij}, \hat{s}_{ij}) \quad (3.73)$$

onde I_{ij}^{noobj} é o complemento de I_{ij}^{obj} e λ_{noobj} é responsável por reduzir a perda ao detectar o fundo. Redmon et al. [2016] atribuem $\lambda_{\text{coord}} = 5$ e $\lambda_{\text{noobj}} = 0,5$.

3. Perda de classificação: avalia o erro na identificação das classes. Se um objeto é detectado, a perda de classificação em cada célula é o erro de entropia cruzada das probabilidades condicionais de classe para cada classe [Redmon e Farhadi, 2018]. Essa perda pode ser definida como:

$$\mathcal{L}_{\text{class}} = \sum_{i=0}^{S^2} I_i^{\text{obj}} \sum_{k \in \text{classes}} \mathcal{L}_{\text{BCE}}(c_{ik}, \hat{c}_{ik}) \quad (3.74)$$

em que I_i^{obj} é um indicador de objeto dado por:

$$I_i^{\text{obj}} = \begin{cases} 1, & \text{se existe um objeto na célula } i \\ 0, & \text{caso contrário} \end{cases} \quad (3.75)$$

3.4.3.4 Arquitetura

Algoritmos de detecção de objetos de última geração baseados em aprendizado profundo normalmente são compostos por um extrator de recursos denominado *backbone*. Tal componente consiste em uma rede convolucional de múltiplas camadas onde são processadas as informações espaciais da imagem e extraídas as características relevantes dos objetos de interesse.

No que se refere ao YOLOv2, o *backbone* é conhecido como Darknet-19 e contempla 19 camadas convolucionais e 5 camadas de agrupamento máximo [Redmon e Farhadi, 2017]. Dentre as camadas convolucionais, 7 aplicam filtros de dimensão 1×1 , de modo que os mapas de recursos resultantes das outras 12 camadas, cujos filtros são 3×3 , sejam comprimidos em profundidade [Lin et al., 2013]. Todas as camadas de agrupamento máximo, por sua vez, precedem normalização em lote e ativação LReLU, bem como utilizam janelas 2×2 e passo 2.

Logo após o extrator de recursos, 4 camadas convolucionais são incorporadas na arquitetura da rede para detecção, sendo 3 camadas com 1024 filtros 3×3 e uma última com $K(5+C)$ filtros 1×1 . Além disso, duas camadas de concatenação são introduzidas em meio às camadas convolucionais do *backbone*, permitindo que mapas de recursos de camadas iniciais sejam adicionados aos mapas de recursos de camadas finais.

Em prol de maior desempenho, o *backbone* do YOLOv3 foi elaborado segundo uma abordagem híbrida do Darknet-19 com uma rede residual [He et al., 2016]. O Darknet-53, como foi intitulado, possui 53 camadas convolucionais e 23 camadas resi-

duais, onde cada camada de convolução, com exceção da última, é seguida por normalização em lote e ativação do tipo LReLU. Para a detecção, foram adicionadas mais 4 camadas de concatenação, 23 de convolução e 2 de reamostragem por um fator de $2\times$ (*upsampling*) [Redmon e Farhadi, 2018]. Esse conjunto de camadas finais responsáveis por prever as classes e as caixas delimitadoras em um detector de objetos baseado em aprendizado profundo é conhecido como *head*.

Tabela 3.2: Darknet-19 [Redmon e Farhadi, 2017].

Tipo de Camada	Filtros	Tamanho/Passo	Saída
Convocional	32	3×3	224x224
Agrupamento máximo		$2 \times 2 / 2$	112x112
Convocional	64	3×3	112x112
Agrupamento máximo		$2 \times 2 / 2$	56x56
Convocional	128	3×3	56x56
Convocional	64	1×1	56x56
Convocional	128	3×3	56x56
Agrupamento máximo		$2 \times 2 / 2$	28x28
Convocional	256	3×3	28x28
Convocional	128	1×1	28x28
Convocional	256	3×3	28x28
Agrupamento máximo		$2 \times 2 / 2$	14x14
Convocional	512	3×3	14x14
Convocional	256	1×1	14x14
Convocional	512	3×3	14x14
Convocional	256	1×1	14x14
Convocional	512	3×3	14x14
Agrupamento máximo		$2 \times 2 / 2$	7x7
Convocional	1024	3×3	7x7
Convocional	512	1×1	7x7
Convocional	1024	3×3	7x7
Convocional	512	1×1	7x7
Convocional	1024	3×3	7x7

Tabela 3.3: Darknet-53 [Redmon e Farhadi, 2018].

Tipo de Camada	Filtros	Tamanho/Passo	Saída
Convocional	32	3×3	256x256
Convocional	64	$3 \times 3 / 2$	128x128
1x	Convocional	32	1×1
	Convocional	64	3×3
Residual			128x128
	Convocional	128	$3 \times 3 / 2$
2x	Convocional	64	1×1
	Convocional	128	3×3
Residual			64x64
	Convocional	256	$3 \times 3 / 2$
8x	Convocional	128	1×1
	Convocional	256	3×3
Residual			32x32
	Convocional	512	$3 \times 3 / 2$
8x	Convocional	256	1×1
	Convocional	512	3×3
Residual			16x16
	Convocional	1024	$3 \times 3 / 2$
4x	Convocional	512	1×1
	Convocional	1024	3×3
Residual			8x8

Note que tanto o Darknet-19 (Tabela 3.2) quanto o Darknet-53 (Tabela 3.3) apresentam uma camada convolucional a menos do que foi mencionado. Ambos os *backbones* foram inicialmente propostos como classificadores para o desafio ImageNet [Russakovsky et al., 2015] e, portanto, a última camada de cada uma delas tinha como princípio modelar um problema de 1000 classes. Assim, de modo a torná-las extratores de características para um detector de objetos, suas últimas camadas convolucionais foram removidas para integração das demais camadas de detecção.

As predições do YOLOv3 são então realizadas em três escalas diferentes, isto é, aplicando $[B(5 + C)]$ filtros de detecção 1×1 em mapas de recursos de três tamanhos diferentes em três locais diferentes da rede (Figura 3.28). Isso permite que as camadas responsáveis pela reamostragem concatenadas com as camadas anteriores obtenham informações semânticas mais significativas dos recursos atualizados e informações mais refinadas do mapa de recursos anterior, melhorando a detecção de pequenos objetos [Lin et al., 2017].

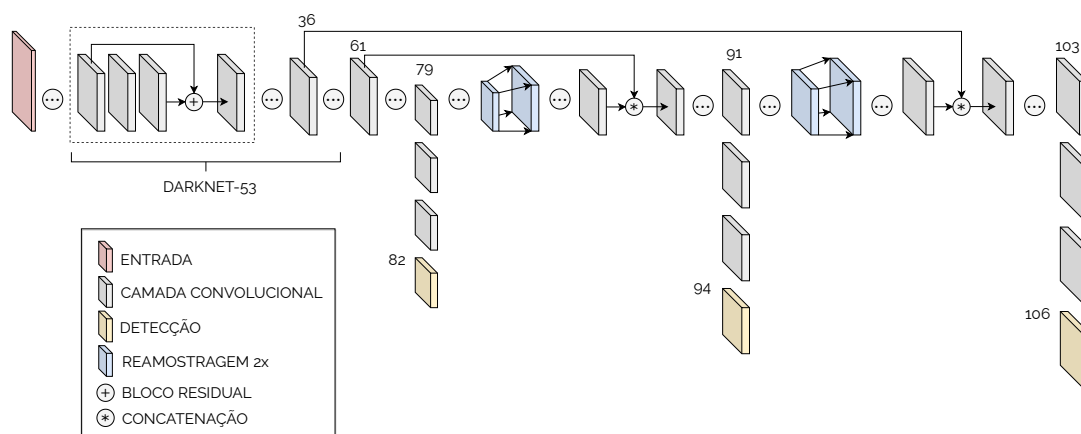


Figura 3.28: Arquitetura simplificada do detector de objetos YOLOv3. Adaptado de Mostofa et al. [2020].

3.4.3.5 Estado da arte

Recentemente, uma série de melhorias no YOLOv3 originou uma quarta versão do YOLO (YOLOv4) [Bochkovskiy et al., 2020]. Dentre as principais novidades que, além de adequarem melhor o treinamento da rede em uma única máquina com GPU, permitiram também aumentar aproximadamente 10% de mAP@0,50 na base de dados COCO e cerca de 12% da taxa de quadros por segundo (FPS) em relação à versão anterior, pode-se destacar a utilização de técnicas dos grupos *bag of freebies* e *bag of specials*.

Os métodos *bag of freebies* são comumente empregados para aperfeiçoar o processo de otimização dos parâmetros da rede. Ao preço de um custo adicional no treinamento, os detectores de objetos podem aumentar consideravelmente seu desempenho sem acrescentar processamento computacional na detecção [Zhang et al., 2019]. Desse grupo, o YOLOv4 considera especialmente técnicas de aumento de dados, como *Cut-Mix* [Yun et al., 2019], *Mosaic* e *self-adversarial training* (SAT) [Bochkovskiy et al., 2020], agendador cossenóide da taxa de aprendizado, regularização *DropBlock* [Ghiasi et al., 2018], suavização de rótulos de classe (*class label smoothing*) [Szegedy et al., 2016] e otimização de hiperparâmetros por algoritmos genéticos [Jocher et al., 2020].

Além disso, a perda de localização do YOLO, que era tradicionalmente calculada pelo MSE nas versões anteriores, agora é definida pela função de perda *Complete IoU* (CIoU) [Zheng et al., 2020]. Essa técnica, que também faz parte do *bag of freebies*, considera simultaneamente a área de sobreposição entre a caixa predita e a caixa real, as proporções de tela (*aspect ratios*) e a distância entre os pontos centrais dessas caixas para garantir maior velocidade de convergência e precisão no problema de regressão das coordenadas das caixas delimitadoras.

Já o grupo *bag of specials* compreende métodos de pós-processamento e módulos de extensão (*plug-in*) que aumentam ligeiramente o custo computacional de inferência em prol de melhorias no desempenho. De modo geral, os métodos de pós-processamento são usados para refinar os resultados da detecção do modelo, como é o caso do *Distance IoU* (DIoU) NMS [Zheng et al., 2020], uma versão aprimorada da supressão não-máxima considerada nas versões anteriores do YOLO. Já os módulos de extensão permitem aumentar o campo receptivo das camadas convolucionais, bem como combinar mapas de recursos provenientes de diferentes partes da rede.

A arquitetura do YOLOv4 dispõe de 110 camadas convolucionais, onde 107 são seguidas por um novo tipo de normalização em lote conhecida como normalização em lote de iteração cruzada (CBN - *cross-iteration batch normalization*) [Yao et al., 2020] e por ativações, sendo 35 pela função LReLU e 72 pela função *Mish*. O *backbone* utilizado para extração de características, denominado CSPDarknet-53, é composto pela integração do Darknet-53 com um módulo *cross stage partial network* (CSPNet), o qual permite que a rede alcance uma combinação de gradiente mais rica, enquanto reduz o custo computacional demandado para detecção [Wang et al., 2020].

A *head* do YOLOv4 é a mesma utilizada no YOLOv3, entretanto, três módulos de extensão são introduzidos entre a *head* e o *backbone*. Esse conjunto de módulos, intitulado *neck*, tem como objetivo agregar informações espaciais e semânticas mais ricas na detecção, onde mapas de recursos vizinhos vindos do fluxo ascendente e descendente da rede são concatenados antes de serem propagados para a *head* (Figura 3.29).

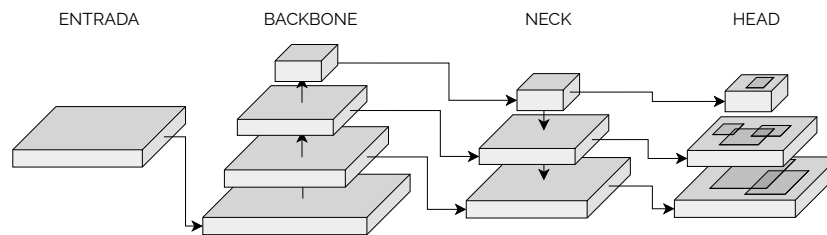


Figura 3.29: Em alto nível, a arquitetura do detector de objetos YOLOv4 pode ser ilustrada como a associação de três principais componentes: (i) *backbone*, (ii) *neck* e (iii) *head*. Adaptado de Bochkovskiy et al. [2020].

O primeiro módulo que compõe o *neck* é o *spatial pyramid pooling* (SPP), cujo propósito é remover a restrição de tamanho fixo da rede, evitando que a imagem de entrada sofra algum tipo de distorção ou recorte para ser propagada [He et al., 2015]. O segundo módulo é o *path aggregation network* (PAN), que é responsável por preservar informações espaciais com precisão através da agregação de parâmetros de diferentes níveis do *backbone* para diferentes níveis da *head* [Liu et al., 2018]. Por fim, tem-se o

spatial attention module (SAM) para identificar qual a parte mais informativa de cada mapa de recursos, de modo que pesos sejam atribuídos a diferentes regiões espaciais de acordo com sua contribuição para a detecção [Woo et al., 2018]. Tanto o módulo PAN quanto o SAM foram ainda ligeiramente modificados para compor a arquitetura do YOLOv4 (Figura 3.30).

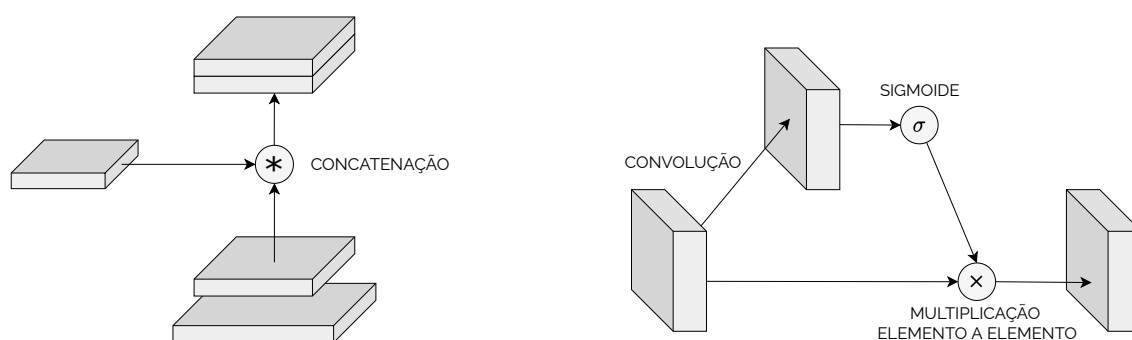


Figura 3.30: Ao contrário da proposta inicial do módulo PAN (à esquerda), a conexão de salto foi substituída por uma operação de concatenação. No módulo SAM (à direita), as camadas de agrupamento máximo e agrupamento médio que precedem a convolução na implementação original foram removidas. Adaptado de Bochkovskiy et al. [2020].

Essa arquitetura complexa da rede YOLOv4, no entanto, requer 59,57 bilhões de FLOPs (BFLOPs) por propagação direta em uma única imagem RGB de tamanho 416×416 pixels. Como essa alta demanda computacional exige GPU para treinamento e detecção em tempo real, os autores também propuseram o Tiny YOLOv4, uma versão mais leve e mais rápida, que requer apenas 6,789 BFLOPs por propagação direta em uma imagem com as mesmas propriedades, cerca de 88,66% a menos que o YOLOv4. Todavia, essa redução de processamento vem ao custo de um pior desempenho.

3.5 Conclusões do capítulo

Este capítulo sintetizou os conceitos essenciais para o entendimento do detector de objetos escolhido para desempenhar a tarefa de detecção de incêndios: a quarta versão do algoritmo *You Only Look Once* (YOLOv4). Para esse fim, foram discutidos princípios básicos de aprendizado de máquina, que serão relevantes para a interpretação dos resultados experimentais posteriormente no Capítulo 5, fundamentos da rede neural artificial e suas principais limitações no processamento digital de imagens e o referencial teórico acerca da rede neural convolucional, topologia que integra a arquitetura do YOLO e que, dado seu alto custo computacional, será otimizada no presente trabalho.

Capítulo 4

Metodologia

Este capítulo apresenta uma nova base de dados contendo imagens rotuladas de fogo e fumaça, a qual será empregada para treinamento do YOLOv4 no problema de detecção de incêndios. Ademais, serão introduzidas técnicas de remoção de filtros em redes neurais convolucionais, com o objetivo de otimizar o desempenho do detector de incêndios para execução em dispositivos de processamento limitado.

4.1 Base de dados

O crescimento da Internet e a popularização da tecnologia da informação, consequências da Indústria 4.0, proporcionaram um aumento exponencial no volume de dados produzidos pela sociedade nas últimas décadas [Obitko e Jirkovskỳ, 2015]. Tal fenômeno desencadeou gradativamente uma série de metodologias ágeis para auxiliar o agrupamento desses dados em conjuntos úteis.

Em harmonia com esse paradigma, redes neurais convolucionais apresentam uma tendência crescente de desempenho à medida que lhes são fornecidos mais exemplos [Mikołajczyk e Grochowski, 2018]. É a partir de variações de cenário, ângulo e iluminação que o modelo identifica padrões recorrentes e adquire conhecimento suficiente para discriminar as classes de interesse em diversas circunstâncias.

4.1.1 Coleta de imagens

Perante o exposto, bem como diante da escassez, retratada na Tabela 4.1, de conjuntos disponíveis na literatura para a tarefa de detecção de incêndios, especificamente, propõe-se um novo conjunto de dados para esse fim. Inicialmente, o processo de coleta de imagens, exclusivamente no espaço de cores RGB, priorizou a Internet e baseou-se nas classes fogo e fumaça. Um cenário com presença de qualquer uma dessas classes requer, no mínimo, atenção, pois pode representar uma situação explícita de emergência. As imagens obtidas foram categorizadas em:

1. fogo: as imagens apresentam apenas ocorrências de fogo.
2. fumaça: as imagens apresentam apenas ocorrências de fumaça.
3. fogo e fumaça: as imagens possuem ocorrências de ambas as classes.
4. fundo: as imagens não contêm ocorrências de nenhuma das classes, mas apresentam objetos ou ambientes que podem ser facilmente confundidos com fogo ou fumaça por um ser humano.

Tabela 4.1: Bases de dados para reconhecimento de fogo e fumaça. A atribuição da tarefa correspondente a cada base foi fundamentada conforme o tipo de rótulo disponível com os dados.

Base de dados	Tarefa	Quantidade	Objeto
Center for Wildfire Research (CWR) [2010]	Classificação, Segmentação	6 Vídeos, 98 Imagens	Fumaça, Fundo
State Key Lab of Fire Science (SKLFS) [2012]	Classificação	6 Vídeos, 81.312 Imagens*	Fumaça, Fundo
Laboratorio di Macchine Intelligenti per il riconoscimento di Video, Immagini e Audio (MIVIA) [2015]	Classificação	31 Vídeos	Fumaça, Fundo
Chino et al. [2015]	Classificação, Segmentação	466 Imagens	Fogo, Fundo
National Fire Research Laboratory (NFRL) [2019]	Classificação	73 Vídeos	Fogo

* 74.989 imagens são de fundo e 6.323 imagens são de fumaça.

A busca por imagens e quadros de vídeos teve como principal alvo *sites* com um notável volume de dados: Pexels, Pixabay, Flickr, Unsplash e Google Imagens. Em todas estas fontes, a pesquisa utilizou o filtro de licença de uso e um conjunto amplo de palavras-chave, contendo termos em português e inglês, como mostrado na Figura 4.1. O critério de escolha foi definido pela diversidade de imagens relevantes em cada resultado da pesquisa.

Apesar dos *sites* serem distintos, há casos em que buscas sobre assuntos similares resultam em imagens iguais. Para evitar imagens duplicadas no conjunto de dados, a ferramenta *Duplicate Image Finder* [Lundrigan, 2018] foi utilizada. A partir dela é possível comparar as imagens com uma função hash e excluir as repetidas. Os tamanhos da imagem e do arquivo são desconsiderados e a tabela de associação é construída conforme os pixels das imagens. Essa propriedade permite identificar imagens duplicadas que foram ligeiramente rotacionadas, por exemplo.

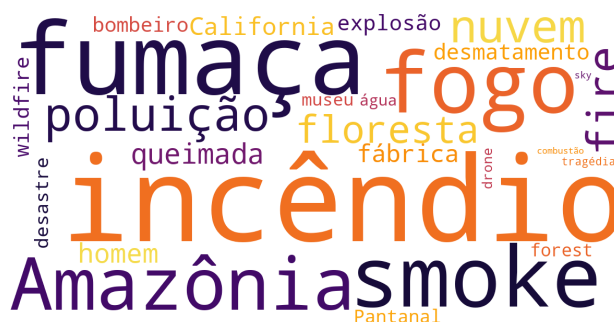


Figura 4.1: Nuvem de palavras contendo a lista de vocábulos utilizados na busca por imagens de fogo e fumaça. Os tamanhos de cada palavra indicam suas respectivas frequências de ocorrência nas legendas das imagens.

4.1.2 Cenários

Identificar as possíveis causas de incêndio dentre as existentes é um fator de grande importância para o aprendizado do algoritmo. A busca pela diversidade de cenários confere ao conjunto de dados maior representatividade das classes e, portanto, um melhor ajuste de parâmetros. Assim, a aquisição das imagens contemplou tanto causas naturais quanto causas antrópicas acidentais e propositais.

Um tipo de ocorrência muito comum é o fogo em vegetações, como plantações e florestas. Em geral, os focos de incêndio começam em regiões limítrofes e se expandem rapidamente para o interior das matas. As fumaças apresentam formatos cônicos bastante característicos de incêndios a longa distância (Figura 4.2).



Figura 4.2: Incêndio em grandes áreas florestais (à esquerda) e queimada não controlada oriunda do manejo de pastagens para gado (à direita).

Incidentes causados por falhas em instalações elétricas ou negligência humanas, tais como incêndios urbanos e acidentes de trânsito, também constituíram grande parcela da base de dados (Figura 4.3). Nesses eventos, em particular, as imagens apresentaram menores resoluções, visto que elas são usualmente registradas por dispositivos móveis de cidadãos comuns.



Figura 4.3: Incêndio resultante de um acidente de trânsito (à esquerda) e incêndio residencial provocado por falhas na instalação elétrica (à direita).

Além de diversificar os cenários, diferentes ângulos também foram considerados através de imagens aéreas capturadas por drones e aviões de combate a incêndio (Figura 4.4). O objetivo é representar o máximo possível das variâncias intraclasse de fogo e fumaça, isto é, as variações visuais entre múltiplas observações de cada classe [Pilarczyk e Skarbek, 2019] e, assim, permitir que os incêndios sejam detectados independentemente da posição da câmera.

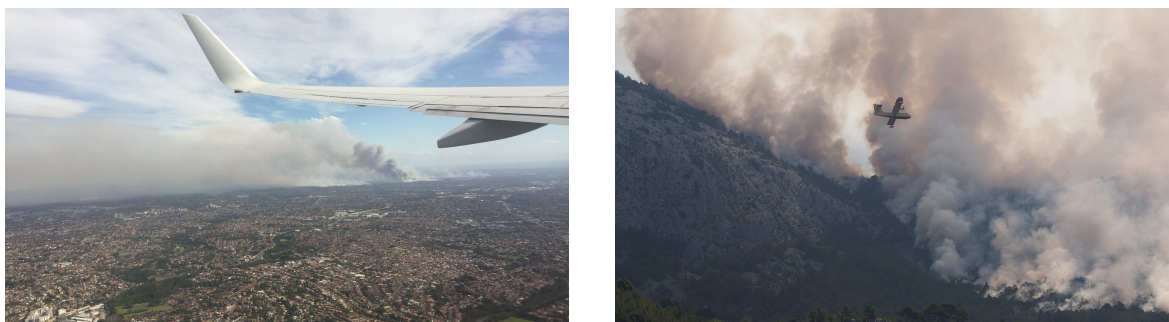


Figura 4.4: Incêndio registrado por avião comercial (à esquerda) e drone de combate a incêndios (à direita).

A identificação das ocorrências de fogo e fumaça presentes nas imagens nem sempre é trivial. Em algumas situações, essas classes podem ser confundidas com outros objetos bastante comuns em cenas monitoradas por câmeras de vigilância, resultando em erros de detecção recorrentes. Dada a similaridade com a fumaça, nuvens e neblinas têm alta probabilidade de provocarem falsos alarmes. O fogo, por sua vez, pode ser constantemente confundido com iluminação noturna ou com reflexos de luz solar [Saeed et al., 2019]. Manter imagens como as da Figura 4.5 no conjunto de dados aumenta a probabilidade do algoritmo aprender essa tênue diferença visual entre os objetos comuns citados e os objetos das classes de interesse e, por consequência, identificar apenas os reais focos de incêndio.



Figura 4.5: Cenários desafiadores. Reflexos solares incidindo sobre a câmera (à esquerda). Fumaça cônica pequena no lado inferior direito da imagem, que é ofuscada pela nuvem cujo formato é similar (à direita).

4.1.3 Geração de imagens

Após a busca por imagens relevantes na Internet, foram identificadas deficiências na composição dos dados. Alguns padrões de interesse apresentaram um pequeno volume de imagens, de modo a desbalancear as classes em determinados cenários. Os padrões de menor incidência foram referentes a focos de incêndio distantes da câmera, fumaças esparsas e fumaças camufladas por vegetações.

Três abordagens foram propostas para resolver essa escassez. A primeira consistiu em conduzir simulações de incêndio e registrá-las conforme a demanda [Toulouse et al., 2017]. Em seguida, imagens foram criadas por meio de colagens utilizando uma ferramenta de edição de fotos [Goyal et al., 2017a]. E por fim, imagens foram obtidas de três câmeras de vigilância instaladas na cidade de Belo Horizonte, Minas Gerais, aumentando o número de exemplos positivos e negativos do banco de dados com cenários reais típicos do monitoramento ambiental.

4.1.3.1 Simulações de incêndio

As simulações de incêndio foram legalmente conduzidas em áreas verdes do Parque Tecnológico de Belo Horizonte (BH-TEC) e consistiu basicamente no processo de atear fogo em folhas secas. Foram registrados 41 vídeos de ocorrências de incêndio no total, com variações de ângulo, distância e luminosidade. A dinâmica das gravações permitiu que 151 quadros fossem extraídos dos vídeos e incorporados ao conjunto de imagens (Figura 4.6). As deficiências pertinentes às fumaças esparsas e as fumaças camufladas pelo ambiente foram resolvidas.



Figura 4.6: Registro de fumaça parcialmente camuflada pela vegetação (à esquerda) e registro de foco esparso integralmente camuflado pela vegetação (à direita).

4.1.3.2 Imagens sintéticas

Imagens sintéticas foram geradas por meio de um software de edição de imagens. O procedimento teve início na separação de padrões de fumaça com cores e formas variadas, conforme exemplificado na Figura 4.7.



Figura 4.7: Padrões de fumaça artificial.

Em seguida, cenários de vegetação e céus nublados foram capturados para compor os fundos da imagem. A densidade de nuvens no céu teve o propósito de induzir a diferenciação com fumaça durante o aprendizado e, assim, reduzir eventuais falsos positivos. Por último, os padrões selecionados com fundo transparente foram distribuídos nas paisagens de forma aleatória, variando o número de ocorrências entre 0 e 4 fumaças e produzindo 95 imagens ao todo (Figura 4.8).



Figura 4.8: Exemplo de imagens sintéticas produzidas com dois focos (à esquerda) e quatro focos (à direita).

4.1.3.3 Imagens de câmeras de vigilância

Imagens obtidas de câmeras de vigilância permitem incorporar no aprendizado do modelo incertezas intrínsecas aos cenários reais, como condições de iluminação e condições climáticas adversas. Essa etapa é primordial para a familiarização do sistema com o ambiente de produção e, por isso, foram adicionados à base de dados exemplos positivos (com fogo ou fumaça) e exemplos negativos (sem fogo ou fumaça) de dois lugares que estão sendo monitorados atualmente no projeto P&D ANEEL D0619 da Companhia Energética de Minas Gerais (CEMIG) pela plataforma “Apaga o Fogo!”¹: (i) BH-TEC e (ii) Parque Estadual Serra Verde, conforme exposto na Figura 4.9.

O edifício institucional do BH-TEC é cercado por uma vasta área verde de preservação ambiental que envolve regiões limítrofes às matas da Universidade Federal de Minas Gerais (UFMG), sendo um lugar bastante propício para a aquisição de imagens. O Parque Estadual Serra Verde, por sua vez, é vítima de centenas de incêndios ao ano. Sua cobertura é feita por duas câmeras separadas por cerca de 100 metros instaladas no terraço do Edifício Gerais da Cidade Administrativa de Minas Gerais (CAMG). Todas as câmeras estão conectadas a um servidor localizado no Centro de Estudos da Fala, Acústica, Linguagem e Música (CEFALA) da UFMG através de um enlace de internet por uma rede privada virtual (VPN - *virtual private network*), promovendo um gerenciamento eficaz das imagens coletadas.



Figura 4.9: Câmera Bosch AUTODOME IP 5000 instalada em um poste no terraço do prédio institucional do BH-TEC (à esquerda). Duas câmeras Bosch NBN-832V DINION IP instaladas no terraço do Edifício Gerais da CAMG (à direita).

4.1.4 Rotulação das imagens

O processo de rotulação das imagens para uma tarefa de detecção de objetos consiste na especificação dos rótulos de classe e das caixas delimitadoras dos objetos. Desse modo, cada imagem obtida para composição da base de dados teve todas as

¹Disponível em <https://apagaofogo.eco.br/>.

suas ocorrências de fogo e fumaça manualmente rotuladas pelos pesquisadores Pedro Vinícius (autor deste trabalho) e Lucas Bispo da empresa *Gaia, solutions on demand*.

A ferramenta utilizada para rotular as caixas delimitadoras de cada classe foi a *BBox Label Tool Multiclass* [Jiaxin Gu, 2017]. Cada imagem rotulada gera um arquivo de texto contendo as coordenadas das caixas delimitadoras dos objetos de interesse. Cada linha do arquivo representa um objeto na imagem correspondente e apresenta o seguinte padrão:

`<xmin> <ymin> <xmax> <ymax> <classe>`

onde `<xmin>` e `<ymin>` são as coordenadas de mínimo da caixa (ponto superior esquerdo), `<xmax>` e `<ymax>` são as coordenadas de máximo da caixa (ponto inferior direito) e `<classe>` é o nome da classe do objeto.

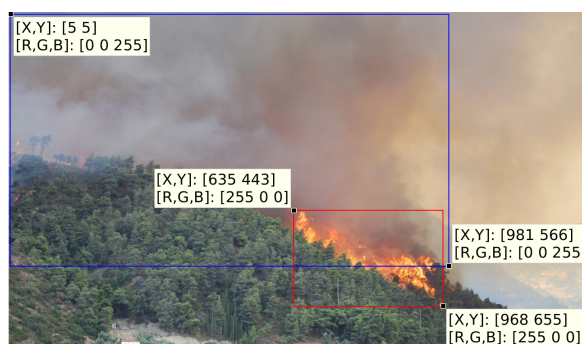


Figura 4.10: Exemplo de rotulação de uma imagem com fogo (caixa vermelha) e fumaça (caixa azul). O arquivo texto associado a essa imagem contém duas linhas: “5 5 981 566 fumaça” e “635 443 968 655 fogo”.

Posteriormente, as coordenadas das caixas delimitadoras de cada imagem são normalizadas no intervalo $[0,1]$ e o nome da classe do objeto é convertido para um identificador de classe, onde 0 indica fumaça e 1 indica fogo.

4.1.5 Estatísticas

O nome dado à base de dados foi D-Fire [Gaia, 2018]. Atualmente são 21.527 imagens rotuladas, cuja proporção por categoria pode ser vista na Tabela 4.2. Apesar da categoria de imagens contendo somente fogo ser uma minoria expressiva na base de dados, o número de ocorrências de fogo em cada imagem que o contém (categorias “fogo” e “fogo e fumaça”) normalmente é alto, com uma média de 2,52 ocorrências por imagem. Já o número médio de fumaças nas categorias “fumaça” e “fogo e fumaça” é de 1,13 ocorrências por imagem.

Tabela 4.2: Distribuição de imagens por categoria na base de dados D-Fire.

Categoria	# Imagens
Fogo	1.164
Fumaça	5.867
Fogo e fumaça	4.658
Fundo	9.838

Uma provável justificativa para essa disparidade de ocorrências por observação é que a aquisição das imagens priorizou incêndios de média e longa distância, onde é possível visualizar vários exemplos com múltiplos focos de fogo. No total são 26.557 marcações de caixas delimitadoras, sendo 14.692 de fogo e 11.865 de fumaça.

4.2 Poda de filtros convolucionais

Redes neurais convolucionais, em sua essência, são modelos profundos que consomem muita memória e demandam muito orçamento computacional. Embora tais particularidades possam ser parcialmente contornadas pelo uso massivo de computação paralela e armazenamento em nuvem, esse tipo de tecnologia em grandes proporções pode ter um custo financeiro muito elevado, principalmente no que se refere ao processo de implantação desses modelos em campo.

Sob essa circunstância, diversas abordagens surgiram para podar as arquiteturas das CNNs, a fim de torná-las escaláveis para dispositivos móveis e máquinas desprovidas de unidades de processamento gráfico [Li et al., 2016; He et al., 2017; Jordao et al., 2020b]. Em síntese, essas técnicas eliminam filtros convolucionais que são redundantes ou que não contribuem efetivamente com a qualidade da saída, de modo a reduzir o número de parâmetros da rede sem uma ou qualquer diminuição abrupta de desempenho. A escolha por eliminar especificamente os filtros convolucionais provém de que, no geral, cerca de 99% das operações de ponto flutuante de uma rede são provenientes das camadas convolucionais [Han et al., 2015].

Dada uma camada convolucional l , o procedimento de poda de filtros dessa camada pode ser sumarizado conforme as etapas a seguir [Li et al., 2016]:

1. Calcule a importância s_j de cada um dos $n^{[l+1]}$ filtros $\mathcal{F}_j^{[l]}$ do conjunto $\mathcal{F}^{[l]} \in \mathbb{R}^{n^{[l+1]} \times n^{[l]} \times k^{[l]} \times k^{[l]}}$ da camada l a partir de uma técnica pré-estabelecida.
2. Ordene os filtros convolucionais conforme suas respectivas importâncias s_j .

3. Remova $p\%$ dos filtros com menor importância e seus respectivos mapas de recursos. Note ainda que a profundidade do conjunto de filtros na camada $l + 1$ também é alterada nesse processo. Essa relação de dimensões entre camadas sucessivas pode ser melhor compreendida na Figura 4.11.

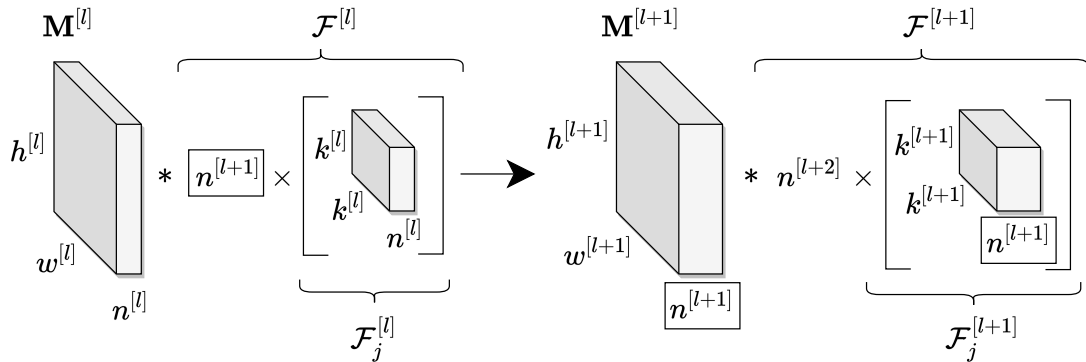


Figura 4.11: A remoção de um filtro $\mathcal{F}_j^{[l]}$ dentre os $n^{[l+1]}$ filtros da camada l implica diretamente na profundidade do conjunto de mapas de recursos $\mathbf{M}^{[l+1]}$ e na profundidade do conjunto de filtros $\mathcal{F}^{[l+1]}$.

Após a remoção de uma certa quantidade de filtros em todas as camadas possíveis da rede, os filtros remanescentes podem apresentar um comportamento indesejável, visto que sua contribuição com a saída pode estar altamente correlacionada com algum filtro que foi removido. Diante disso, é aconselhável que se faça um reajuste desses parâmetros por meio do *fine-tuning*. A capacidade que a rede tem de recuperar sua acurácia após esse processo é chamada de plasticidade [Mittal et al., 2018].

Todas as etapas mencionadas constituem uma única iteração do método de poda. Caso seja necessário realizar mais uma iteração, a rede podada na iteração i é dada como entrada para o método na iteração $i + 1$, conforme exemplificado na Figura 4.12. Isso permite que a poda seja feita tanto gradualmente até atingir $p\%$ de remoção (*iterative pruning*) quanto diretamente removendo $p\%$ dos filtros em uma única iteração (*single pruning*). O método iterativo, no entanto, não será considerado no presente trabalho, dada a alta demanda computacional para realizar todas as etapas várias vezes.

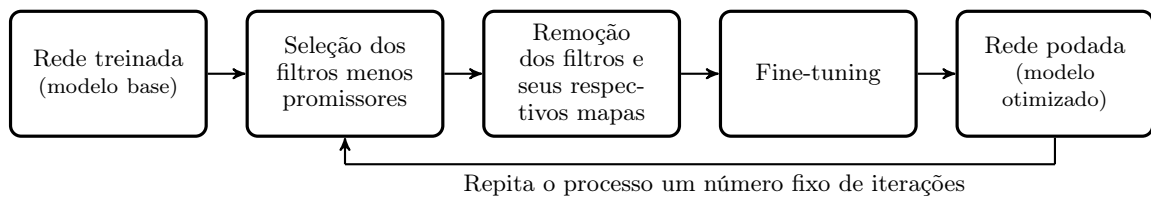


Figura 4.12: Processo iterativo de poda de filtros em redes neurais convolucionais.

4.2.1 Técnicas baseadas em critérios de importância

As técnicas de poda se diferem principalmente quanto à quantificação da importância de cada filtro convolucional para o desempenho final da rede. Diversos critérios foram propostos para identificar os filtros menos promissores, tais como ativações nulas (APoZ) [Hu et al., 2016], baixa entropia [Luo e Wu, 2017], normas ℓ_1 [Han et al., 2015; Li et al., 2016] ou ℓ_2 [He et al., 2018; Lin et al., 2018] pequenas. No entanto, métodos baseados essencialmente em critérios de importância (*importance-based criterion*) podem apresentar resultados similares dada a elevada plasticidade das redes neurais profundas [Mittal et al., 2018]. Outro problema dessas técnicas é que elas olham para cada filtro individualmente, ou seja, não consideram as interações entre os filtros convolucionais de uma mesma camada e nem os seus respectivos efeitos na saída da rede. Um filtro que é pouco relevante para o desempenho final pode melhorar significativamente a precisão se for usado em conjunto com um ou mais filtros complementares. Em contrapartida, um filtro particularmente relevante pode se tornar redundante ou até mesmo prejudicial para o desempenho da rede quando usado com outros filtros [Guyon e Elisseff, 2003].

4.2.2 Técnicas baseadas em múltiplas projeções

Métodos baseados em projeções de recursos em espaços latentes de baixa dimensão modelam tipicamente a relação entre variáveis dependentes e independentes. No problema particular da detecção de incêndios, deseja-se capturar a relação das saídas dos filtros convolucionais (variáveis independentes) com os rótulos das classes fogo e fumaça (variáveis dependentes) e, a partir dessa relação, remover $p\%$ dos filtros convolucionais menos discriminativos em cada camada.

4.2.2.1 Representações das variáveis

Com esse intuito, cada imagem de treinamento é propagada pela arquitetura da rede, de modo a produzir um mapa de recursos para cada um dos filtros convolucionais. Dado que os mapas de recursos apresentam alta dimensionalidade, a operação de agrupamento *global max pooling* é empregada para reduzir a dimensão de cada mapa a um único valor. Ao contrário da operação *max pooling* que extrai valores máximos em pequenas regiões do mapa, o filtro dessa operação tem tamanho equivalente ao tamanho do mapa de recursos e, como resultado, extrai apenas a *feature* de maior valor, isto é, a característica mais evidente daquela saída, conforme ilustrado na Figura 4.13.

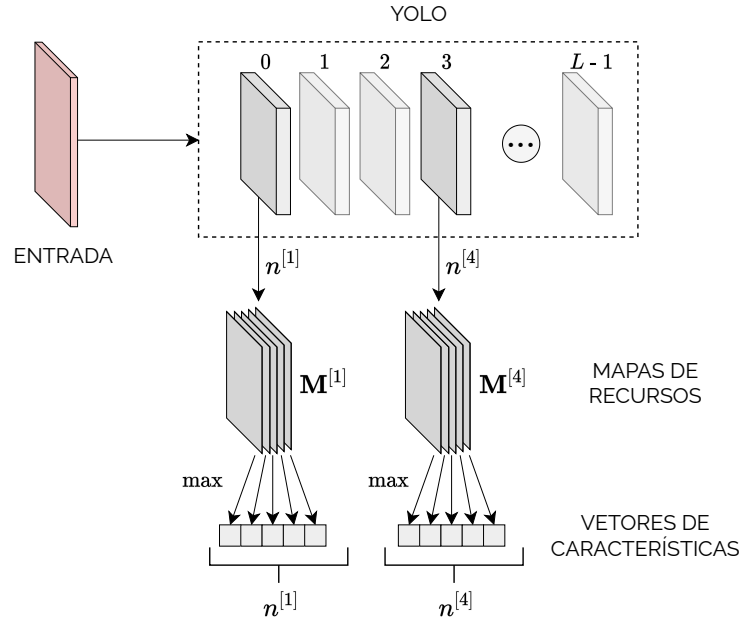


Figura 4.13: Representação das saídas dos filtros convolucionais (mapas de recursos) como vetores de características, sendo esses utilizados para projeções em espaços de baixa dimensão (espaços latentes). Nessa figura, os mapas de recursos com $n^{[l+1]}$ canais foram retratados como $n^{[l+1]}$ fatias de profundidade unitária. Além disso, os volumes de maior opacidade exemplificam camadas cujos filtros podem ser removidos pela metodologia descrita na Seção 4.2 sem que ocorram problemas na saída da rede.

Observe ainda que a camada convolucional l da rede com $n^{[l+1]}$ filtros produz $n^{[l+1]}$ mapas de recursos de tamanho $h^{[l+1]} \times w^{[l+1]}$ e esses mapas são reduzidos em um único vetor de características de tamanho $n^{[l+1]}$. Supondo um conjunto de treinamento com m imagens, tem-se m vetores de características de tamanho $n^{[l+1]}$ e, por consequência, uma matriz de variáveis independentes $\mathbf{X}^{[l]} \in \mathbb{R}^{m \times n^{[l+1]}}$ para cada uma das l_p camadas convolucionais cujos filtros podem ser removidos.

A etapa de representação de filtros descrita foi definida em [Jordao et al., 2020b] para problemas de classificação e pode ser integralmente empregada em detectores de objetos baseados em redes convolucionais, como o YOLO. Entretanto, a representação dos rótulos no trabalho de Jordao et al. [2020b] não é diretamente escalável para a detecção de objetos. Em um problema de classificação com C classes, cada imagem de entrada está associada a um único rótulo de classe como saída e as variáveis dependentes podem ser representadas por uma matriz de classe binária $\mathbf{Y} \in \mathbb{B}^{m \times C}$, cujos valores são dados por:

$$y_{ij} = \begin{cases} 1, & \text{se o rótulo da imagem } i \text{ é a classe } j \\ 0, & \text{caso contrário} \end{cases} \quad (4.1)$$

Já em uma tarefa de detecção de objetos, uma imagem pode conter múltiplos objetos de classes distintas e, portanto, uma entrada pode apresentar mais de um rótulo de classe. Por exemplo, a primeira imagem da base de dados pode conter duas ocorrências de fogo e a segunda imagem, por sua vez, pode conter uma ocorrência de fogo e duas de fumaça. Em vista disso, as variáveis dependentes foram definidas segundo duas estratégias: (i) modelagem binária e (ii) modelagem multiclasse.

A modelagem binária agrega os rótulos de fogo e fumaça ($C = 2$) em um único rótulo equivalente, que pode ser interpretado como uma classe incêndio ($C = 1$). Nesse caso, a saída é representada por um vetor de classe binária $\mathbf{Y} \in \mathbb{B}^{m \times 1}$, de modo que os valores são determinados pelas seguintes condições:

$$y_i = \begin{cases} 1, & \text{se a imagem } i \text{ possui fogo ou fumaça} \\ 0, & \text{caso contrário} \end{cases} \quad (4.2)$$

A modelagem multiclasse, no entanto, trata os rótulos de classe separadamente e os atribui conforme a área de incidência de cada classe na imagem. Se apenas uma das classes está presente na imagem, o rótulo dessa classe é atribuído à imagem. Caso as duas classes estejam presentes na mesma imagem, o rótulo da classe que compreende o maior número de pixels na cena é atribuído à imagem. Sejam n_i e m_i os números de focos de fumaça e fogo presentes na imagem i , respectivamente, a representação das variáveis dependentes tem princípio em um vetor real $\mathbf{y} \in \mathbb{R}^{m \times 1}$ com valores estabelecidos por:

$$y_i = \begin{cases} 1, & \text{se } \sum_{q=1}^{m_i} A_q^{(\text{fogo})} < \sum_{p=1}^{n_i} A_p^{(\text{fumaça})} \\ 2, & \text{se } \sum_{q=1}^{m_i} A_q^{(\text{fogo})} \geq \sum_{p=1}^{n_i} A_p^{(\text{fumaça})} \\ 0, & \text{se } m_i = n_i = 0 \end{cases} \quad (4.3)$$

onde $A_q^{(\text{fogo})}$ indica a área da caixa delimitadora que compreende a q -ésima ocorrência de fogo na imagem e $A_p^{(\text{fumaça})}$ a área da caixa delimitadora que compreende a p -ésima ocorrência de fumaça na imagem. A primeira condição indica o evento em que a soma das áreas de todas as ocorrências de fogo na imagem i é menor que a soma das áreas de todas as ocorrências de fumaça na mesma imagem e, portanto, considera a fumaça como o rótulo da imagem ($y_i = 1$). A segunda condição indica o evento oposto, onde a soma das áreas de todas as ocorrências de fogo é maior ou igual que a soma das áreas de todas as ocorrências de fumaça. Nesse cenário, a classe fogo é definida como rótulo da imagem ($y_i = 2$). Por fim, a terceira condição indica uma imagem sem ocorrências de nenhuma das classes, ou seja, ausência de fogo e fumaça ($y_i = 0$).

As variáveis dependentes \mathbf{y} são então transformadas em uma matriz binária $\mathbf{Y} \in \mathbb{B}^{m \times (C+1)}$ através da codificação esparsa *one-hot encoding*. Em outras palavras, o rótulo que indica a ausência de fogo e fumaça na imagem i ($y_i = 0$) é codificado em $\mathbf{y}_i = [1 \ 0 \ 0]$, o rótulo que indica maior presença de fumaça na imagem i ($y_i = 1$) é codificado em $\mathbf{y}_i = [0 \ 1 \ 0]$ e o rótulo que indica maior presença de fogo na imagem i ($y_i = 2$) é codificado em $\mathbf{y}_i = [0 \ 0 \ 1]$.

4.2.2.2 PLS-VIP

O próximo passo consiste em projetar individualmente cada uma das l_p variáveis independentes em um espaço latente de dimensão r (múltiplas projeções). Com essa finalidade, a técnica de poda PLS-VIP [Jordao et al., 2020b] utiliza o método de projeção de recursos discriminativos *Partial Least Squares* (PLS), que decompõe $\mathbf{X}^{[l]} \in \mathbb{R}^{m \times n^{[l+1]}}$ e $\mathbf{Y} \in \mathbb{B}^{m \times (C+1)}$ centralizados em torno da média, conforme:

$$\begin{aligned} \mathbf{X}^{[l]} &= \mathbf{TP}^T + \mathbf{E} \\ \mathbf{Y} &= \mathbf{UQ}^T + \mathbf{F} \end{aligned} \quad (4.4)$$

onde \mathbf{T} e $\mathbf{U} \in \mathbb{R}^{m \times r}$ são, respectivamente, matrizes de entrada e saída de pontuação no espaço latente (*scores*), $\mathbf{P} \in \mathbb{R}^{n^{[l+1]} \times r}$ e $\mathbf{Q} \in \mathbb{R}^{(C+1) \times r}$ são, respectivamente, as matrizes de entrada e saída de carregamento (*loadings*) e $\mathbf{E} \in \mathbb{R}^{m \times n^{[l+1]}}$ e $\mathbf{F} \in \mathbb{R}^{m \times (C+1)}$ são resíduos. Note, portanto, que a modelagem de \mathbf{Y} nesse caso é multiclasse.

Segundo essa decomposição, o PLS estima uma matriz de projeção $\widetilde{\mathbf{W}}^{[l]} \in \mathbb{R}^{n^{[l+1]} \times r}$ para redução da dimensionalidade de $\mathbf{X}^{[l+1]}$ de $\mathbb{R}^{m \times n^{[l+1]}}$ para $\mathbb{R}^{m \times r}$, tal que $r \ll n^{[l+1]}$. Cada componente $\widetilde{\mathbf{w}}_j^{[l]} \in \widetilde{\mathbf{W}}^{[l]} = [\widetilde{\mathbf{w}}_1^{[l]}, \widetilde{\mathbf{w}}_2^{[l]}, \dots, \widetilde{\mathbf{w}}_r^{[l]}]$ representa a máxima covariância entre as variáveis $\mathbf{X}^{[l]} \widetilde{\mathbf{w}}$ e \mathbf{Y} :

$$\begin{aligned} \widetilde{\mathbf{w}}_j^{[l]} &= \arg \max_{\widetilde{\mathbf{w}}} \left(\text{cov}(\mathbf{X}^{[l]} \widetilde{\mathbf{w}}, \mathbf{Y}) \right) \\ &\text{sujeito a: } \|\widetilde{\mathbf{w}}\| = 1 \end{aligned} \quad (4.5)$$

tal que a covariância entre as variáveis $\mathbf{X}^{[l]} \widetilde{\mathbf{w}}$ e \mathbf{Y} é dada por:

$$\text{cov}(\mathbf{X}^{[l]} \widetilde{\mathbf{w}}, \mathbf{Y}) = \frac{1}{m-1} (\mathbf{X}^{[l]} \widetilde{\mathbf{w}})^T \mathbf{Y} \quad (4.6)$$

O problema de otimização formalizado por (4.5) pode ser resolvido através dos algoritmos *singular value decomposition* (SVD) e *nonlinear iterative partial least squares* (NIPALS) [Wold, 1973]. Em conformidade com Jordao et al. [2020b], o presente trabalho considera o algoritmo NIPALS para esse fim. Enquanto o SVD encontra to-

das as $n^{[l+1]}$ componentes, o NIPALS obtém apenas as r primeiras componentes, sendo então consideravelmente mais rápido. O Algoritmo 1 apresenta as etapas do NIPALS, onde a convergência é dada quando não ocorrem mais mudanças em $\tilde{\mathbf{w}}_j^{[l]}$ ou quando um número máximo de iterações n_{\max} é atingido [Jordao et al., 2020b].

Algoritmo 1 NIPALS para PLS.

Entrada: $\mathbf{X}^{[l]} \in \mathbb{R}^{m \times n^{[l+1]}}$, $\mathbf{Y} \in \mathbb{B}^{m \times (C+1)}$, $r \in \{1, \dots, n^{[l+1]}\}$

Saída: $\tilde{\mathbf{W}}^{[l]} \in \mathbb{R}^{n^{[l+1]} \times r}$

- 1: **para** $j = 1$ até r **faça**
 - 2: Inicialize aleatoriamente $\mathbf{u}_j \in \mathbb{R}^{m \times 1}$
 - 3: **enquanto** $\tilde{\mathbf{w}}_j^{[l]}$ sofre alterações e n_{\max} não é atingido **faça**
 - 4: $\tilde{\mathbf{w}}_j^{[l]} = \frac{\mathbf{X}^{[l]T} \mathbf{u}_j}{\|\mathbf{X}^{[l]T} \mathbf{u}_j\|}$
 - 5: $\mathbf{t}_j = \mathbf{X}^{[l]} \tilde{\mathbf{w}}_j^{[l]}$
 - 6: $\mathbf{q}_j = \frac{\mathbf{Y}^T \mathbf{t}_j}{\|\mathbf{Y}^T \mathbf{t}_j\|}$
 - 7: $\mathbf{u}_j = \mathbf{Y} \mathbf{q}_j$
 - 8: **fim enquanto**
 - 9: $\mathbf{p}_j = \mathbf{X}^{[l]T} \mathbf{t}_j$
 - 10: $\mathbf{X}^{[l]} = \mathbf{X}^{[l]} - \mathbf{t}_j \mathbf{p}_j^T$
 - 11: $\mathbf{Y} = \mathbf{Y} - \mathbf{t}_j \mathbf{q}_j^T$
 - 12: **fim para**
-

Após a obtenção da matriz de projeção $\tilde{\mathbf{W}}^{[l]}$ pelo Algoritmo 1 para cada uma das l_p variáveis independentes, o próximo passo é estimar a medida de contribuição de cada filtro convolucional de uma determinada camada para o espaço latente. Para isso, emprega-se a métrica *variable importance in projection* (VIP) [Mehmoed et al., 2012], cuja importância $f_k^{[l]}$ de um filtro $\mathcal{F}_k^{[l]}$ é calculada por:

$$f_k^{[l]} = \sqrt{n^{[l+1]} \frac{\sum_{j=1}^r S_j \left(\frac{\tilde{w}_{jk}^{[l]}}{\|\tilde{\mathbf{w}}_j^{[l]}\|^2} \right)}{\sum_{j=1}^r S_j}} \quad (4.7)$$

onde o termo S_j representa a soma dos quadrados explicados pelo j -ésimo componente ($\mathbf{q}_j^2 \mathbf{t}_j^T \mathbf{t}_j$) e $\tilde{w}_{jk}^{[l]} / \|\tilde{\mathbf{w}}_j^{[l]}\|^2$ a importância do filtro k da camada l .

Finalmente, serão descartados $p\%$ dos filtros com menor pontuação $f_k^{[l]}$ em cada uma das l_p camadas convolucionais. Seja $l^* \in \mathbb{Z}^{l_p \times 1}$ os índices de todas as camadas que terão seus filtros removidos, tem-se uma remoção de $\sum_{l \in l^*} \lfloor \frac{p}{100} n^{[l+1]} \rfloor$ filtros ao todo na arquitetura. A rede resultante é submetida ao processo de *fine-tuning* para reajustar

os valores dos filtros que permaneceram.

4.2.2.3 CCA-CV

Ao passo que o PLS visa maximizar a covariância entre as combinações lineares das variáveis independentes e dependentes, a análise em componentes canônicas (CCA - *canonical correlation analysis*) busca encontrar r pares de projeções lineares $\widetilde{\mathbf{W}}_{\mathbf{x}} \in \mathbb{R}^{n^{[l+1]} \times r}$ e $\widetilde{\mathbf{W}}_{\mathbf{y}} \in \mathbb{R}^{C \times r}$ tal que as correlações entre $\widetilde{\mathbf{W}}_{\mathbf{x}}^T \mathbf{X}^{[l]T}$ e $\widetilde{\mathbf{W}}_{\mathbf{y}}^T \mathbf{Y}^T$ sejam maximizadas. Dado o j -ésimo par de vetores canônicos $\widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]} \in \widetilde{\mathbf{W}}_{\mathbf{x}}^{[l]} = [\widetilde{\mathbf{w}}_{\mathbf{x}_1}^{[l]}, \widetilde{\mathbf{w}}_{\mathbf{x}_2}^{[l]}, \dots, \widetilde{\mathbf{w}}_{\mathbf{x}_r}^{[l]}]$ e $\widetilde{\mathbf{w}}_{\mathbf{y}_j}^{[l]} \in \widetilde{\mathbf{W}}_{\mathbf{y}}^{[l]} = [\widetilde{\mathbf{w}}_{\mathbf{y}_1}^{[l]}, \widetilde{\mathbf{w}}_{\mathbf{y}_2}^{[l]}, \dots, \widetilde{\mathbf{w}}_{\mathbf{y}_r}^{[l]}]$, o CCA pode ser modelado como:

$$\begin{aligned} \widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]}, \widetilde{\mathbf{w}}_{\mathbf{y}_j}^{[l]} &= \arg \max_{\widetilde{\mathbf{w}}_{\mathbf{x}}, \widetilde{\mathbf{w}}_{\mathbf{y}}} \left(\text{corr}(\widetilde{\mathbf{w}}_{\mathbf{x}} \mathbf{X}^{[l]T}, \widetilde{\mathbf{w}}_{\mathbf{y}} \mathbf{Y}^T) \right) \\ &\text{sujeito a: } \|\widetilde{\mathbf{w}}_{\mathbf{x}}\| = \|\widetilde{\mathbf{w}}_{\mathbf{y}}\| = 1 \end{aligned} \quad (4.8)$$

onde a correlação entre as variáveis $\widetilde{\mathbf{w}}_{\mathbf{x}} \mathbf{X}^{[l]T}$ e $\widetilde{\mathbf{w}}_{\mathbf{y}} \mathbf{Y}^T$ é dada por:

$$\text{corr}(\widetilde{\mathbf{w}}_{\mathbf{x}} \mathbf{X}^{[l]T}, \widetilde{\mathbf{w}}_{\mathbf{y}} \mathbf{Y}^T) = \frac{\text{cov}(\widetilde{\mathbf{w}}_{\mathbf{x}} \mathbf{X}^{[l]T}, \widetilde{\mathbf{w}}_{\mathbf{y}} \mathbf{Y}^T)}{\sqrt{\text{cov}(\widetilde{\mathbf{w}}_{\mathbf{x}} \mathbf{X}^{[l]T}, \widetilde{\mathbf{w}}_{\mathbf{x}} \mathbf{X}^{[l]T})} \sqrt{\text{cov}(\widetilde{\mathbf{w}}_{\mathbf{y}} \mathbf{Y}^T, \widetilde{\mathbf{w}}_{\mathbf{y}} \mathbf{Y}^T)}} \quad (4.9)$$

As etapas do NIPALS para obter as primeiras r componentes canônicas na formulação do problema de otimização (4.8) é apresentada no Algoritmo 2.

Algoritmo 2 NIPALS para CCA.

Entrada: $\mathbf{X}^{[l]} \in \mathbb{R}^{m \times n^{[l+1]}}$, $\mathbf{Y} \in \mathbb{R}^{m \times C}$, $r \in \{1, \dots, \min(n^{[l+1]}, C)\}$

Saída: $\widetilde{\mathbf{W}}_{\mathbf{x}}^{[l]} \in \mathbb{R}^{n^{[l+1]} \times r}$, $\widetilde{\mathbf{W}}_{\mathbf{y}}^{[l]} \in \mathbb{R}^{C \times r}$

- 1: $\mathbf{X}'^{[l]} = \mathbf{X}^{[l]} (\mathbf{X}^{[l]T} \mathbf{X}^{[l]})^{-\frac{1}{2}}$
 - 2: $\mathbf{Y}' = \mathbf{Y} (\mathbf{Y}^T \mathbf{Y})^{-\frac{1}{2}}$
 - 3: $\mathbf{A} = \mathbf{X}'^{[l]T} \mathbf{Y}'$
 - 4: **para** $j = 1$ até r **faça**
 - 5: Inicialize $\widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]}$ com as médias das colunas de \mathbf{A}
 - 6: **enquanto** $\widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]}$ sofre alterações e n_{\max} não é atingido **faça**
 - 7: $\widetilde{\mathbf{w}}_{\mathbf{y}_j}^{[l]} = \frac{\mathbf{A}^T \widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]}}{\|\mathbf{A}^T \widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]}\|}$
 - 8: $\widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]} = \frac{\mathbf{A} \widetilde{\mathbf{w}}_{\mathbf{y}_j}^{[l]}}{\|\mathbf{A} \widetilde{\mathbf{w}}_{\mathbf{y}_j}^{[l]}\|}$
 - 9: **fim enquanto**
 - 10: $\mathbf{A} = \mathbf{A} - (\widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]T} \mathbf{A} \widetilde{\mathbf{w}}_{\mathbf{y}_j}^{[l]}) \widetilde{\mathbf{w}}_{\mathbf{x}_j}^{[l]} \widetilde{\mathbf{w}}_{\mathbf{y}_j}^{[l]T}$
 - 11: **fim para**
-

De maneira análoga ao PLS-VIP, propõe-se o CCA-CV, uma estratégia de poda baseada em múltiplas projeções obtidas com o CCA. Uma vez que a matriz de variáveis dependentes \mathbf{Y} foi modelada como um vetor coluna ($C = 1$), as l_p projeções são realizadas considerando apenas uma única componente:

$$\begin{aligned} r &\in \{1, \dots, \min(n^{[l+1]}, C)\} \\ r &\in \{1, \dots, \min(n^{[l+1]}, 1)\} \\ n^{[l+1]} &> 1, r \in \{1, 1\} \\ \text{logo } r &= 1 \end{aligned} \tag{4.10}$$

Como consequência, cada matriz de projeção $\widetilde{\mathbf{W}}_{\mathbf{x}}^{[l]}$ será composta por apenas um vetor canônico $\widetilde{\mathbf{w}}_{\mathbf{x}}^{[l]} \in \mathbb{R}^{n^{[l+1]} \times 1}$, no qual cada valor \widetilde{w}_k desse vetor representará o grau de correlação do filtro $\mathcal{F}_k^{[l]}$ com a saída. Assim, $p\%$ dos filtros com os menores valores absolutos $|\widetilde{w}_k|$ de cada uma das l_p camadas são descartados e, posteriormente, os filtros restantes são ajustados através do *fine-tuning*.

4.2.3 Técnicas baseadas em agrupamento

Uma outra classe de técnicas de poda que consideram as interações entre os filtros convolucionais são as baseadas em agrupamento, onde os filtros de uma mesma camada convolucional são reunidos por similaridade em um número de conjuntos previamente determinado. A finalidade do agrupamento é identificar justamente os filtros que são mais similares, isto é, que compõem o mesmo grupo e, assim, removê-los da camada. A premissa assumida aqui é que filtros similares produzem respostas similares e, portanto, são redundantes, contribuindo pouco ou até atrapalhando a saída final da rede.

4.2.3.1 Agrupamento hierárquico aglomerativo

Dada uma camada convolucional l com $n^{[l+1]}$ filtros, o primeiro passo da técnica de poda por agrupamento proposta consiste em compor uma matriz de similaridades $\mathbf{S} \in \mathbb{R}^{n^{[l+1]} \times n^{[l+1]}}$ a partir de todas as possíveis comparações par a par dos filtros. A similaridade entre um filtro $\mathcal{F}_u^{[l]}$ e um filtro $\mathcal{F}_v^{[l]}$ é dada pelo coeficiente de correlação de Pearson:

$$\text{corr}(\mathcal{F}_u^{[l]}, \mathcal{F}_v^{[l]}) = \frac{\text{cov}(\mathcal{F}_u^{[l]}, \mathcal{F}_v^{[l]})}{\sqrt{\sigma^2(\mathcal{F}_u^{[l]})\sigma^2(\mathcal{F}_v^{[l]})}} \tag{4.11}$$

tal que os filtros tridimensionais são transformados em um vetor unidimensional (*flattening*) e que σ^2 é a variância dos valores do filtro.

Os filtros achatados são então agrupados segundo o algoritmo de agrupamento hierárquico aglomerativo (HAC - *hierarchical agglomerative clustering*) com o critério de ligação completo (*complete-linkage*), assim como no trabalho de Ghosh et al. [2019]. A matriz de distâncias utilizada para agrupar os filtros é definida como o oposto da matriz de similaridades ($\mathbf{D} = 1 - \mathbf{S}$) e o número de *clusters* como a quantidade de filtros que se deseja manter na camada l , isto é, $\lfloor (1 - p)n^{l+1} \rfloor$ grupos.

Após o agrupamento, cada grupo compreenderá essencialmente filtros semelhantes, enquanto que filtros muito distintos serão dispostos em grupos diferentes. Os grupos com mais de um filtro, portanto, passarão a sugerir redundância e os filtros pertencentes a esses grupos serão considerados potenciais candidatos para remoção.

A fim de identificar os filtros que serão removidos, cada filtro pertencente a grupos com mais de um filtro terá sua correlação com os demais grupos mensurada. A correlação do v -ésimo filtro do grupo i , isto é, filtro $\widehat{\mathcal{F}}_{iv}^{[l]}$, com um grupo $\widehat{\mathcal{F}}_j^{[l]} \subset \mathcal{F}^{[l]}$ contendo n' filtros é dada por:

$$\text{corr}(\widehat{\mathcal{F}}_{iv}^{[l]}, \widehat{\mathcal{F}}_j^{[l]}) = \max \left(\{ |\text{corr}(\widehat{\mathcal{F}}_{iv}^{[l]}, \widehat{\mathcal{F}}_{j1}^{[l]})|, \dots, |\text{corr}(\widehat{\mathcal{F}}_{iv}^{[l]}, \widehat{\mathcal{F}}_{jn'}^{[l]})| \} \right) \quad (4.12)$$

e a similaridade desse filtro com todos os filtros dos outros grupos, ou seja, com os filtros que não são tão similares a ele, é dada por:

$$\text{sim}(\widehat{\mathcal{F}}_{iv}^{[l]}) = \max \left(\{ |\text{corr}(\widehat{\mathcal{F}}_{iv}^{[l]}, \widehat{\mathcal{F}}_1^{[l]})|, \dots, |\text{corr}(\widehat{\mathcal{F}}_{iv}^{[l]}, \widehat{\mathcal{F}}_g^{[l]})| \} \right) \quad (4.13)$$

onde g é o número de grupos que não compreendem o filtro $\widehat{\mathcal{F}}_{iv}^{[l]}$, isto é, $g = \lfloor n^{l+1}(1 - p) \rfloor - 1$ grupos.

Assim, apenas o filtro menos similar de cada grupo com mais de um filtro será mantido, de tal forma que se tenha apenas um filtro por grupo. Como consequência desse método, intitulado de HAC-PC, $p\%$ dos filtros menos diversificados em cada camada da rede serão eliminados.

4.3 Conclusões do capítulo

Este capítulo introduziu a base de dados D-Fire para detecção de fogo e fumaça, bem como descreveu todo o seu processo de elaboração. As imagens foram coletadas, em sua grande maioria, da Internet, considerando uma variedade de cenários, iluminações e ângulos. Entretanto, a escassez de registros de incêndios com determinadas particularidades requisitaram esforços adicionais na aquisição de imagens através de ensaios em campo, monitoramento de áreas verdes por câmeras de vigilância e elaboração

de imagens sintéticas por uma ferramenta gráfica.

A fim de reduzir o custo computacional do YOLOv4 treinado no conjunto de dados descrito, também foram apresentadas três classes de técnicas de poda de filtros convolucionais, uma baseada em critérios de importância, outra baseada em múltiplas projeções em espaços latentes de baixa dimensão e, por fim, uma baseada em agrupamento por similaridades. Tendo em vista que essas duas últimas vertentes lidam não apenas com as particularidades de cada filtro mas também com a interação entre filtros de uma mesma camada, o que pode ser bastante efetivo, foram apresentadas duas técnicas de múltiplas projeções (PLS-VIP e CCA-CV) e uma de agrupamento (HAC-PC). A técnica PLS-VIP foi proposta especificamente para a tarefa de classificação de imagens e precisou ser adaptada para a tarefa de detecção de objetos. As técnicas CCA-CV e HAC-PC, por sua vez, foram elaboradas na presente pesquisa.

Capítulo 5

Resultados

Este capítulo apresenta o procedimento realizado para treinamento do detector de objetos YOLOv4 no conjunto de imagens D-Fire, bem como a escolha dos seus respectivos hiperparâmetros referentes ao aprendizado e à inferência. Posteriormente, o modelo obtido é submetido a uma etapa de redução de parâmetros, onde são empregadas as técnicas de poda de filtros convolucionais propostas. Os resultados experimentais são então reportados e comparados aos obtidos por outras abordagens comumente utilizadas na literatura. Finalmente, um estudo de caso é realizado para análise do detector de incêndios otimizado em um sistema embarcado com recursos limitados.

5.1 Configuração experimental

A máquina utilizada para conduzir todos os experimentos, com exceção do estudo de caso, foi uma Intel(R) Xeon(R) Gold 6140 de 18 núcleos e 36 threads operando a 2.30 GHz, 256GB de RAM, RAID 5 com SSDs de 1TB e Placa de Vídeo NVIDIA Quadro P5000 com 2560 cores CUDA e 16GB GDDR5X, executando o sistema operacional Ubuntu 18.04.3 LTS e a ferramenta de compilação CUDA 10.1, V10.1.243.

No que se refere à implementação dos algoritmos, cada treinamento e *fine-tuning* foi efetuado em linguagem C com o código-fonte do *framework* Darknet [Redmon e Bochkovskiy, 2013] e as técnicas de remoção de filtros convolucionais foram implementadas em Python 3.7 com o suporte das bibliotecas PyTorch 1.3.1 [Paszke et al., 2019] e scikit-learn 0.24.1 [Pedregosa et al., 2011].

Por fim, a base de dados foi dividida arbitrariamente em dois conjuntos independentes e identicamente distribuídos, sendo 80% para treinamento (17.221 imagens) e 20% para teste (4.306 imagens). Além disso, técnicas de expansão de dados foram aplicadas aleatoriamente nas imagens do conjunto de treinamento durante o aprendizado dos parâmetros, ou seja, a cada iteração variações de saturação, brilho e matiz em diferentes intensidades foram empregadas no lote de dados atual [Bochkovskiy et al., 2020].

5.2 Hiperparâmetros

A primeira etapa realizada para a elaboração do detector de incêndios robusto e computacionalmente viável para dispositivos de processamento limitado foi a construção de um modelo base a partir do treinamento da rede YOLOv4. Para tal fim, a otimização dos parâmetros foi feita com o algoritmo gradiente descendente em lote com *momentum* $\beta = 0,9$ por 30.000 iterações e com um mini-lote de tamanho $m' = 64$. Cada mini-lote foi ainda subdividido em 16 partições, de modo que apenas $64/16 = 4$ imagens fossem simultaneamente processadas pela GPU de cada vez.

Com o intuito de evitar o *overfitting*, a regularização ℓ_2 com $\lambda = 0,0005$ e o critério de convergência *early stopping point* foram empregados, assim como a técnica *warm-up* com $\kappa = 4$ e $\tau = 1.000$ e o agendador *multistep* com fator de redução $\gamma = 0,1$ nos marcos $\mathcal{M} = \{24.000, 27.000\}$ foram utilizadas para aprimorar a otimização. O tamanho da imagem foi definido em $N = 416$, a dimensão da grade de células $S = N/32 = 13$ e o número de caixas delimitadoras por célula $B = 3$ [Redmon et al., 2016]. A arquitetura Tiny YOLOv4 também foi treinada com as mesmas configurações para estabelecer, posteriormente, comparações com as redes resultantes de diferentes técnicas de poda. Ademais, os pesos das camadas de ambas as redes foram inicializados por transferência de aprendizado com o conjunto de dados ImageNet-1000 [Deng et al., 2009].

Como dito na Seção 3.1.5, a taxa de aprendizado α é um dos hiperparâmetros mais importantes da otimização. Pequenas variações no seu valor podem influenciar bastante o treinamento do modelo e, conseqüentemente, sua generalização. Desse modo, um estudo foi realizado para definir a melhor taxa de aprendizado para cada arquitetura. Os treinamentos com as diferentes taxas de aprendizado foram realizados em uma partição arbitrária de 80% do conjunto de treinamento (13.776 imagens) e a taxa de aprendizado para cada arquitetura foi determinada segundo o maior valor de mAP@0,50 (interpolação em todos os pontos) obtido no conjunto de validação, isto é, nos 20% de dados restantes do conjunto de treinamento (3.445 imagens).

Assim, em conformidade com a Figura 5.1, as taxas de aprendizado determinadas foram $\alpha = 5 \times 10^{-3}$ (mAP@0,50 = 63,97%) para o Tiny YOLOv4 e $\alpha = 1 \times 10^{-3}$ para o YOLOv4 (mAP@0,50 = 76,08%). Todos os demais hiperparâmetros de treinamento da rede reportados foram definidos empiricamente. Desempenhar ajustes individuais para as dezenas de hiperparâmetros existentes nessas arquiteturas ou até mesmo ajustes simultâneos desses hiperparâmetros é custoso computacionalmente e demanda bastante tempo. Cada treinamento das redes Tiny YOLOv4 e YOLOv4 requer, respectivamente, cerca de 8 horas e 50 horas nas condições de máquina especificadas.

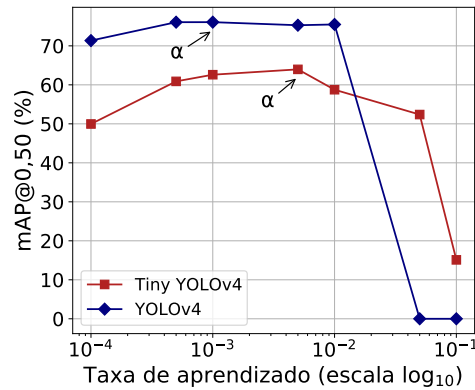


Figura 5.1: Ajuste da taxa de aprendizado das redes YOLOv4 e Tiny YOLOv4 conforme o valor de $mAP@0,50$ no conjunto de validação.

Embora não seja um hiperparâmetro que implica na otimização dos parâmetros, o limiar de confiança t_{conf} afeta diretamente a credibilidade das detecções geradas pelo modelo treinado. Valores muito altos de t_{conf} podem diminuir substancialmente a taxa de verdadeiros positivos e valores muito baixos de t_{conf} podem aumentar consideravelmente a taxa de falsos positivos. Tendo isso em vista, o limiar de confiança de cada rede foi ajustado segundo o processo de validação cruzada k -fold com $k = 5$, onde foram treinados cinco modelos com os hiperparâmetros estabelecidos, cada um avaliado para dez valores de t_{conf} uniformemente distribuídos entre 0 e 0,9 (Figura 5.2).

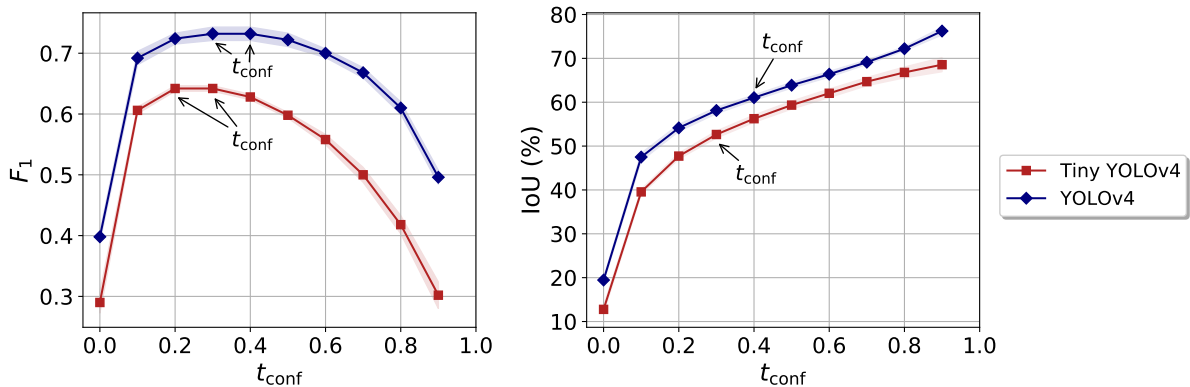


Figura 5.2: Efeito do limiar de confiança nas redes segundo os valores médios de F_1 e IoU nos 5 *folds* destinados à validação. As áreas transparentes ao redor da curva de cada rede especificam os limites inferior e superior obtidos, isto é, um desvio padrão abaixo da média e um desvio padrão acima da média, respectivamente.

O melhor limiar de confiança para cada rede foi definido como sendo o valor que, na média dos cinco modelos, maximiza a medida F_1 e, em caso de empate, apresenta o maior IoU médio em todas as classes. Segundo a medida F_1 , houve um empate

entre dois valores de t_{conf} , tal que os melhores limiares de confiança médios obtidos na validação cruzada *5-fold* para o Tiny YOLOv4 foram 0,2 e 0,3 ($F_1 = 0,642$) e para o YOLOv4 foram 0,3 e 0,4 ($F_1 = 0,732$). Todavia, como o IoU médio cresce com o aumento do limiar de confiança, o maior valor de t_{conf} que maximiza F_1 para cada arquitetura implicará no maior valor de IoU médio também, isto é, $t_{\text{conf}} = 0,3$ para o Tiny YOLOv4 (IoU = 52,64%) e $t_{\text{conf}} = 0,4$ para o YOLOv4 (IoU = 61,04%).

5.3 Análise de desempenho

Em seguida, os cinco modelos de cada uma das redes de detecção obtidos na validação cruzada *5-fold* foram avaliados no conjunto de teste (4.306 imagens), cada rede com o seu melhor limiar de confiança definido anteriormente. Os resultados são apresentados na Tabela 5.1 em função da média e do desvio padrão dos cinco modelos, isto é, $\mu \pm \sigma$, tal que μ é a média e σ é o desvio padrão.

Tabela 5.1: Desempenho das redes Tiny YOLOv4 e YOLOv4 no conjunto de teste.

Rede	mAP@0,50 (%)	AP _{fumaça} (%)	AP _{fogo} (%)	F_1	IoU (%)
Tiny YOLOv4	61,47 \pm 0,65	67,43 \pm 0,90	55,52 \pm 0,60	0,63 \pm 0,01	51,89 \pm 0,59
YOLOv4	73,98 \pm 0,40	80,51 \pm 0,43	67,45 \pm 0,54	0,72 \pm 0,01	60,51 \pm 0,31

Ao comparar os resultados das redes, é possível perceber que o YOLOv4 tem o desempenho médio superior ao do Tiny YOLOv4 em todas as métricas de avaliação consideradas. Esse resultado é esperado, uma vez que a arquitetura do YOLOv4 possui 89 camadas convolucionais a mais e, portanto, é capaz de extrair características mais refinadas das imagens, isto é, constituir representações visuais em níveis de abstração mais complexos. Em outras palavras, a rede YOLOv4 é capaz de detectar mais ocorrências reais de fogo e fumaça (verdadeiros positivos) e menos ocorrências falsas (falsos positivos) do que a rede Tiny YOLOv4, bem como as localizações das ocorrências encontradas são, em média, mais precisas.

Note ainda que ambas as redes têm maior dificuldade em detectar ocorrências de fogo do que ocorrências de fumaça. Uma eventual justificativa para isso é que a marcação manual dos eventos de fogo para treinamento tende a ser mais subjetiva, tal que focos pequenos próximos uns dos outros podem ser rotulados tanto individualmente quanto por uma única caixa delimitadora. Além disso, as ocorrências de fogo presentes nas imagens geralmente estão em estágios iniciais de propagação e, portanto, se tratam de regiões com um menor número de pixels. Assim, o processamento dessas regiões por várias camadas pode diminuir consideravelmente a resolução espacial dos mapas de recursos ao longo da arquitetura e, conseqüentemente, dificultar a visibili-

dade de pequenos focos de incêndio nesses mapas de baixa resolução. Essa limitação dos detectores de objetos de um único estágio, inclusive, vem sendo investigada desde as primeiras arquiteturas propostas e já apresenta uma melhora significativa até o presente momento [Redmon e Farhadi, 2018; Bochkovskiy et al., 2020].

Apesar das métricas de desempenho observadas no conjunto de teste para o YOLOv4 não serem excepcionalmente grandes, isto é, acima de 90%, seus valores devem ser analisados com cautela. As métricas AP@0,50, mAP@0,50 e F_1 , em particular, penalizam todas as ocorrências verdadeiras que eventualmente deixaram de ser detectadas, o que é plausível. No entanto, há uma grande quantidade de exemplos na base de dados que possuem mais de um foco de incêndio por imagem, alguns muito pequenos, inclusive. Deixar de detectar um fogo ou uma fumaça em uma imagem com múltiplas ocorrências afeta significativamente essas métricas, porém pode não ser tão crítico em um monitoramento real com câmera estática de médio alcance, por exemplo. Como nesse contexto o campo de visão da câmera é constante e a distância da câmera para a área monitorada não é tão grande, identificar uma única ocorrência de incêndio já pode ser fundamental para a supressão de múltiplos focos na mesma região.

A fim de certificar tais considerações, a medida F_1 foi calculada novamente no conjunto de teste para os cinco modelos de cada rede obtidos na validação cruzada, porém desconsiderando os múltiplos objetos por cena e, por consequência, suas respectivas localizações. Essa simplificação do problema permitiu simular o desempenho das redes em uma tarefa de classificação, onde os rótulos de classe possíveis para cada imagem foram definidos como: (i) fundo, (ii) fumaça, (iii) fogo e (iv) fogo e fumaça.

Tabela 5.2: Desempenho das redes segundo a medida F_1 no conjunto de teste ao desconsiderar os múltiplos objetos por imagem.

Rede	F_1
Tiny YOLOv4	0,8556 ± 0,0045
YOLOv4	0,9121 ± 0,0029

Com base na Tabela 5.2, os valores de F_1 das redes Tiny YOLOv4 e YOLOv4 aumentaram, em média, cerca de 35,81% e 26,68%, respectivamente, quando comparados aos valores médios de F_1 na tarefa de detecção (Tabela 5.1). Além de evidenciar a maior complexidade do problema de detecção de múltiplos focos de incêndio, esses resultados apontam que mesmo não identificando todas as ocorrências em uma imagem, os detectores treinados são capazes de determinar satisfatoriamente se uma cena tem fogo ou fumaça.

No caso do IoU, valores muito altos indicam que as caixas delimitadoras pre-

ditas pelo modelo coincidem rigorosamente com as caixas delimitadoras definidas no processo de rotulação. Levando em consideração que as classes são muito abstratas e delimitar suas regiões de incidência para treinamento dos modelos é um trabalho manual e relativamente subjetivo, um valor acima de 50% já implica em boas localizações dos eventos [Hoiem et al., 2012].

Para fornecer uma avaliação ainda melhor da qualidade do YOLOv4 na identificação de fogo e fumaça, são apresentadas algumas detecções em exemplos do conjunto de teste. As Figuras 5.3 a 5.7 ilustram detecções bem-sucedidas do modelo em situações reais e sintéticas de incêndio com uma variedade de ângulos, ambientes e iluminações. As Figuras 5.8 a 5.10 mostram alguns dos cenários desafiadores superados pelo modelo, ou seja, cenas com um grande grau de incerteza envolvido, típicas de monitoramento ambiental assistido por câmeras, como fenômenos da natureza e objetos imprevisíveis. Essa robustez evidenciada pode ser atribuída principalmente à base de dados D-Fire, cujas imagens permitem modelar satisfatoriamente a variância intraclasse do fogo e da fumaça e, conseqüentemente, proporcionar à rede um aprendizado generalizado. As Figuras 5.11 e 5.12, por sua vez, apresentam algumas limitações, isto é, falsos positivos e falsos negativos. As caixas delimitadoras verdes e rosas indicam, respectivamente, eventos de fogo (*fire*) e fumaça (*smoke*) e os valores em cada caixa correspondem à pontuação de confiança da rede YOLOv4 na detecção.

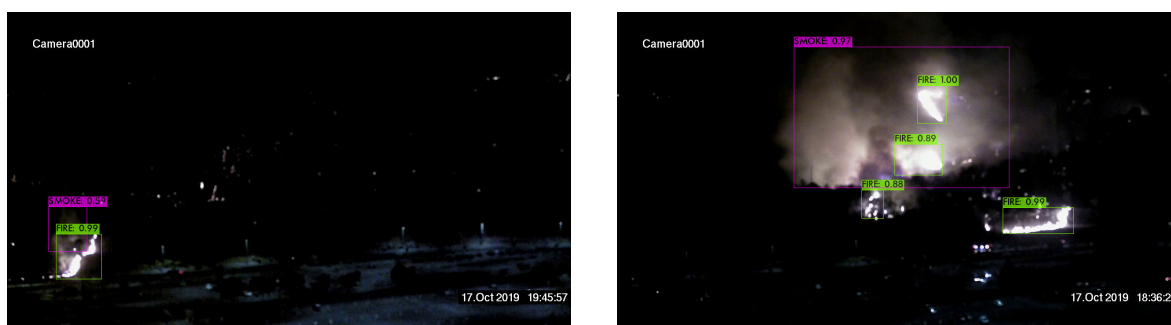


Figura 5.3: Detecções de fogo e fumaça em ambientes noturnos.



Figura 5.4: Detecções de focos de incêndio em estágios iniciais.



Figura 5.5: Detecções de fumaça com padrão cônico.

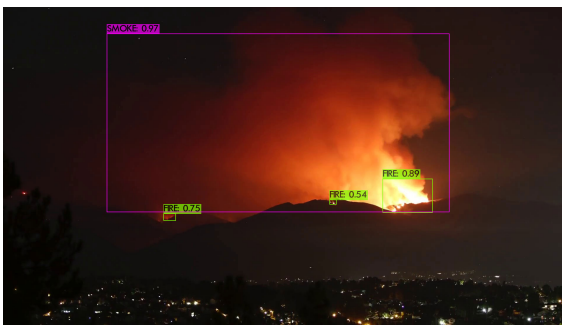


Figura 5.6: Detecções de múltiplos focos de fogo.

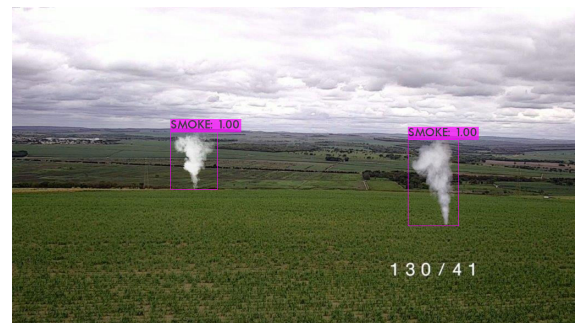


Figura 5.7: Detecção por vista aérea (à esquerda) e de focos sintéticos (à direita).



Figura 5.8: Aranha (à esquerda) e gotas de chuva (à direita) na lente da câmera.

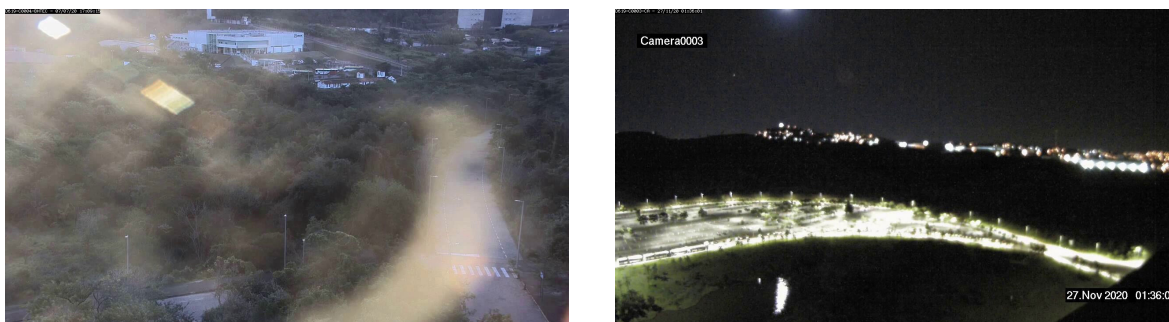


Figura 5.9: Reflexos solares (à esquerda) e luzes artificiais (à direita).

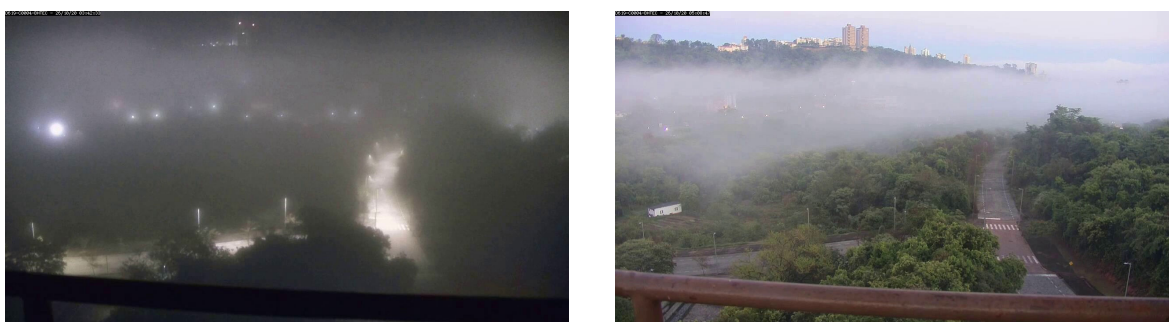


Figura 5.10: Ambiente com névoa (à esquerda) e neblina com aspecto de fumaça (à direita).



Figura 5.11: Nuvens equivocadamente detectadas como fumaça.



Figura 5.12: Fumaças esparsas não detectadas.

5.4 Otimização da rede

O bom desempenho apresentado pelo YOLOv4 incide sobre um dos principais dilemas das redes convolucionais. Arquiteturas mais complexas, isto é, com um maior número de camadas e um maior número de filtros, vêm ao custo de um alto custo computacional, bem como de uma velocidade de detecção reduzida. Enquanto o YOLOv4 processa 47 FPS, o Tiny YOLOv4 processa 342 FPS nas condições de máquina especificadas. Ainda que compatível com a premissa de processamento em tempo real (≥ 30 FPS) [Sadeghi e Forsyth, 2014], isso só é possível com o suporte de processadores extremamente potentes, como as GPUs. Além disso, são necessários 256,04MB de memória externa para armazenar os pesos da arquitetura do YOLOv4, ao passo que para o Tiny YOLOv4 são necessários apenas 23,53MB.

No âmbito do monitoramento ambiental de larga escala, soluções como essa, embora exequíveis, demandam um alto investimento em infraestrutura, seja para aquisição de GPUs ou seja para a contratação de serviços de computação e armazenamento em nuvem. Diante disso, o ideal é que esse problema seja tratado diretamente na construção do modelo de aprendizado profundo. Dado que o YOLOv4 possui um total de 33.917 filtros convolucionais e 63.943.071 parâmetros, assim como seu pipeline de detecção é composto por uma única rede neural com 110 camadas convolucionais, onde apenas as camadas que precedem as 3 camadas de detecção não podem ser modificadas ($l_p = 107$), é possível otimizar sua arquitetura de ponta a ponta a partir da remoção dos filtros menos importantes para a identificação das classes fogo e fumaça.

Por esse motivo, foram investigadas 7 técnicas de poda de filtros convolucionais no detector de incêndios, conforme apresentado na Tabela 5.3. A técnica mais promissora será aquela capaz de promover uma maior compressão na arquitetura sem que haja uma depreciação considerável no desempenho do modelo [Jordao et al., 2020b].

Do ponto de vista da otimização, tem-se um problema biobjetivo, tal que se deseja, simultaneamente, minimizar o custo computacional e maximizar a performance da rede. Em problemas de otimização multiobjetivo, os objetivos são tipicamente conflitantes, isto é, a melhora em um objetivo implica necessariamente na piora de outro e, portanto, não é possível definir uma única solução ótima, mas um conjunto de soluções de compromisso denominado fronteira Pareto-ótima. Essas soluções também são conhecidas como soluções não dominadas, já que não há nenhuma outra solução que as supere em todos os objetivos concomitantemente. Entretanto, o problema de poda de filtros convolucionais também é um problema de natureza combinatória e, na prática, os algoritmos empregados não garantem otimalidade global, mas sim uma possível aproximação da fronteira Pareto-ótima.

Tabela 5.3: Descrição das técnicas de poda de filtros convolucionais consideradas.

Técnica	Descrição
SCRATCH	Remove $p\%$ dos filtros convolucionais de cada camada da rede aleatoriamente. Os filtros remanescentes, no entanto, não são reaproveitados e o processo de <i>fine-tuning</i> é feito a partir do zero (<i>from scratch</i>), isto é, a partir de uma inicialização aleatória dos parâmetros.
RANDOM	Remove $p\%$ dos filtros convolucionais de cada camada da rede aleatoriamente. Nesse caso, o processo de <i>fine-tuning</i> é feito considerando os valores dos parâmetros que permaneceram na arquitetura.
L1	Remove $p\%$ dos filtros convolucionais com menor norma ℓ_1 de cada camada da rede. Os filtros remanescentes são ajustados com <i>fine-tuning</i> .
L2	Remove $p\%$ dos filtros convolucionais com menor norma ℓ_2 de cada camada da rede. Os filtros remanescentes são ajustados com <i>fine-tuning</i> .
PLS-VIP	Converte cada filtro em um vetor de características e os projeta em um espaço latente de dimensão $r = 2$ pelo PLS, que captura a relação entre cada filtro e seu rótulo de classe. Em seguida, remove $p\%$ dos filtros de cada camada com menor pontuação VIP e os filtros restantes são ajustados com <i>fine-tuning</i> .
CCA-CV	Converte cada filtro em um vetor de características e os projeta em um espaço latente de dimensão $r = 1$ pelo CCA, que captura a relação entre cada filtro e seu rótulo de classe. Em seguida, remove $p\%$ dos filtros de menor correlação absoluta com a saída em cada camada da rede e os filtros restantes são ajustados com <i>fine-tuning</i> .
HAC-PC	Agrupa os filtros convolucionais por similaridade em cada camada e remove $p\%$ dos filtros de maior correlação absoluta com os demais grupos (<i>intercluster</i>). Os filtros remanescentes são ajustados com <i>fine-tuning</i> .

Sob essa concepção, uma possível estimativa da fronteira Pareto-ótima obtida por cada uma das técnicas de poda será analisada a partir de agora. Cada solução de uma fronteira se refere a um modelo de rede com $p\%$ dos seus filtros removidos em cada camada. Ao todo foram investigadas 9 taxas de poda distintas e uniformemente distribuídas no intervalo $[10\%, 90\%]$ para compor cada fronteira e as propriedades de custo computacional dos modelos resultantes são mostradas na Tabela 5.4. É interessante ressaltar que como as técnicas de poda eliminam um número constante de filtros por camada, essa relação de compressão com o custo computacional será igual em todas elas. Outro ponto importante a ser mencionado é que remover $p\%$ dos filtros em cada camada da rede não implica necessariamente em eliminar $p\%$ dos parâmetros de toda a arquitetura. Dado que um filtro é um tensor tridimensional e que a rede de detecção YOLOv4 é composta em sua grande maioria por camadas convolucionais, remover

$p\%$ filtros implica em remover mais do que $p\%$ dos parâmetros totais da rede, o que constata que a poda de filtros é realmente uma abordagem bastante promissora para compressão de redes convolucionais profundas.

Tabela 5.4: Relação do custo computacional do modelo com a intensidade de poda empregada na arquitetura da rede.

Filtros removidos	Taxa de poda p (por camada)	Parâmetros removidos	Memória (MB)	BFLOPs ($416 \times 416 \times 3$)
3261	10%	18,81%	207,90	48,610
6581	20%	35,82%	164,37	38,497
9888	30%	50,78%	126,08	29,506
13208	40%	63,80%	92,75	21,758
16576	50%	74,96%	64,18	14,988
19837	60%	83,88%	41,33	9,768
23157	70%	90,91%	23,34	5,561
26464	80%	95,92%	10,49	2,516
29784	90%	98,96%	2,69	0,674

Os mesmos hiperparâmetros do processo de treinamento, definidos na Seção 5.2, foram agora considerados para a etapa de *fine-tuning*, com exceção da taxa de aprendizado, que foi ajustada individualmente para cada técnica de poda. No entanto, como são 7 taxas de aprendizado distintas para cada um dos 9 modelos de cada uma das 7 técnicas de poda, seriam necessários 441 experimentos só para o ajuste da taxa de aprendizado. Em vista disso, a taxa de aprendizado para realizar o *fine-tuning* foi determinada a partir de uma investigação em duas taxas de poda arbitrárias ($p = 30\%$ e $p = 80\%$), tal que a taxa de aprendizado escolhida para cada técnica nesses valores de p será generalizada para todos os modelos de sua respectiva fronteira.

Por meio da Figura 5.13, nota-se que a taxa de aprendizado que maximiza o mAP@0,50 no conjunto de validação para todas as técnicas com ambas taxas de poda consideradas está na região $5 \times 10^{-3} \leq \alpha \leq 10^{-2}$. No que se refere à taxa de poda de 30%, os modelos podados por todas as técnicas tiveram seu desempenho máximo com $\alpha = 5 \times 10^{-3}$. A partir dessa taxa de aprendizado, o desempenho dos modelos assumiu um comportamento decrescente até que se obteve uma completa divergência com $\alpha = 10^{-1}$ (mAP@0,50 = 0,00%). Já para a taxa de poda de 80%, os modelos podados com todas as técnicas tiveram seu desempenho máximo com $\alpha = 10^{-2}$, exceto o modelo podado pela técnica L2, que teve seu desempenho mais elevado com uma taxa de aprendizado substancialmente menor, isto é, $\alpha = 5 \times 10^{-3}$. Além disso, o modelo podado com a técnica L2 também foi o único que divergiu para $\alpha = 10^{-1}$.

De acordo com essa análise, restaram dois valores candidatos para a taxa de

aprendizado: $\alpha = 5 \times 10^{-3}$ e $\alpha = 10^{-2}$. Da teoria exposta na Seção 3.1.5, sabe-se que uma taxa de aprendizado elevada implica em um passo grande em direção ao mínimo e, por consequência, pode induzir o algoritmo de otimização a saltar os ótimos locais, tal como uma taxa de aprendizado baixa, por sua vez, resulta em um passo pequeno em direção ao mínimo e pode reduzir bastante a velocidade de convergência da otimização. Como o *fine-tuning* de cada modelo será efetuado com 30.000 iterações ao invés das 8.000 iterações utilizadas para ajuste da taxa de aprendizado, optou-se pela taxa de aprendizado menor ($\alpha = 5 \times 10^{-3}$) em cada um dos 9 modelos de todas as 7 técnicas. Assim, mesmo que a velocidade de convergência seja inferior à da otimização com taxas de aprendizado maiores, o maior número de iterações permite evitar uma convergência precoce, sem que haja ainda o risco de saltar ótimos locais de boa qualidade.

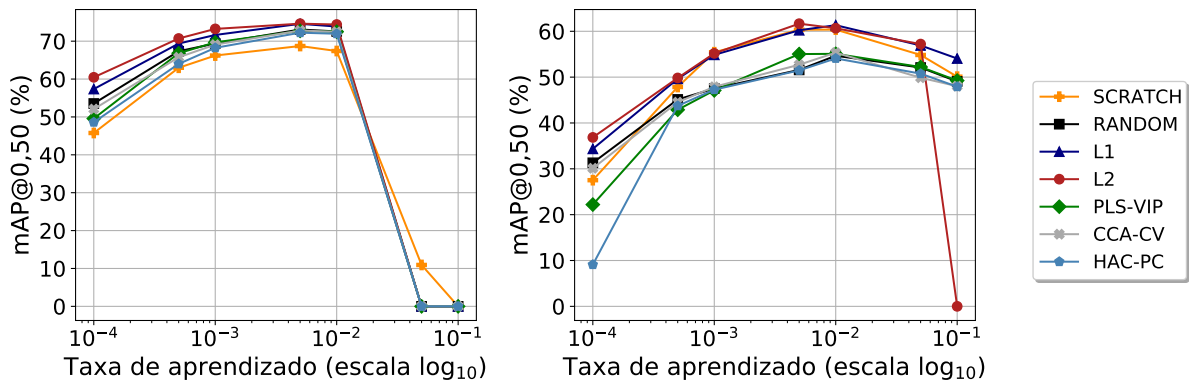


Figura 5.13: Ajuste da taxa de aprendizado dos modelos podados com $p = 30\%$ (à esquerda) e $p = 80\%$ (à direita) conforme o valor de mAP@0,50 no conjunto de validação. Todos os hiperparâmetros foram similares aos definidos no treinamento das redes, porém cada procedimento de *fine-tuning* nesse ajuste da taxa de aprendizado foi feito por 8.000 iterações com um fator de redução $\gamma = 0, 1$ nos marcos $\mathcal{M} = \{6.400, 7.200\}$.

A Figura 5.14 apresenta os modelos otimizados por cada uma das técnicas em curvas, representando assim diferentes compromissos entre o custo computacional em BFLOPs (10^9 FLOPs) e o desempenho segundo as métricas mAP@0,50, AP@0,50, F_1 e IoU no conjunto de teste. Para fins de comparação, os pontos individuais representam ainda o mesmo compromisso para as redes YOLOv4 e Tiny YOLOv4 sem otimização, porém para os desempenhos médios dos cinco modelos obtidos na validação cruzada 5-*fold* e cujos valores estão na Tabela 5.1. Os resultados das otimizações considerando diferentes técnicas também podem ser visualizados nas Tabelas 5.5 a 5.9, onde o melhor valor para cada taxa de poda p é dado em negrito. As colunas com mais valores em negrito, portanto, refletem as técnicas que produziram melhores resultados para um maior número de taxas de poda. Além disso, as avaliações em cada métrica são comparadas com o desempenho médio da rede YOLOv4 sem poda ($p = 0\%$), onde os

símbolos \uparrow , \downarrow e \bullet indicam, respectivamente, melhora, degradação e empate. É importante mencionar também que a rede Tiny YOLOv4 não foi otimizada devido ao seu desempenho substancialmente inferior ao da rede YOLOv4. No entanto, suas avaliações nas métricas consideradas serviram como um limite inferior para cenários cujo processamento computacional é mais restrito.

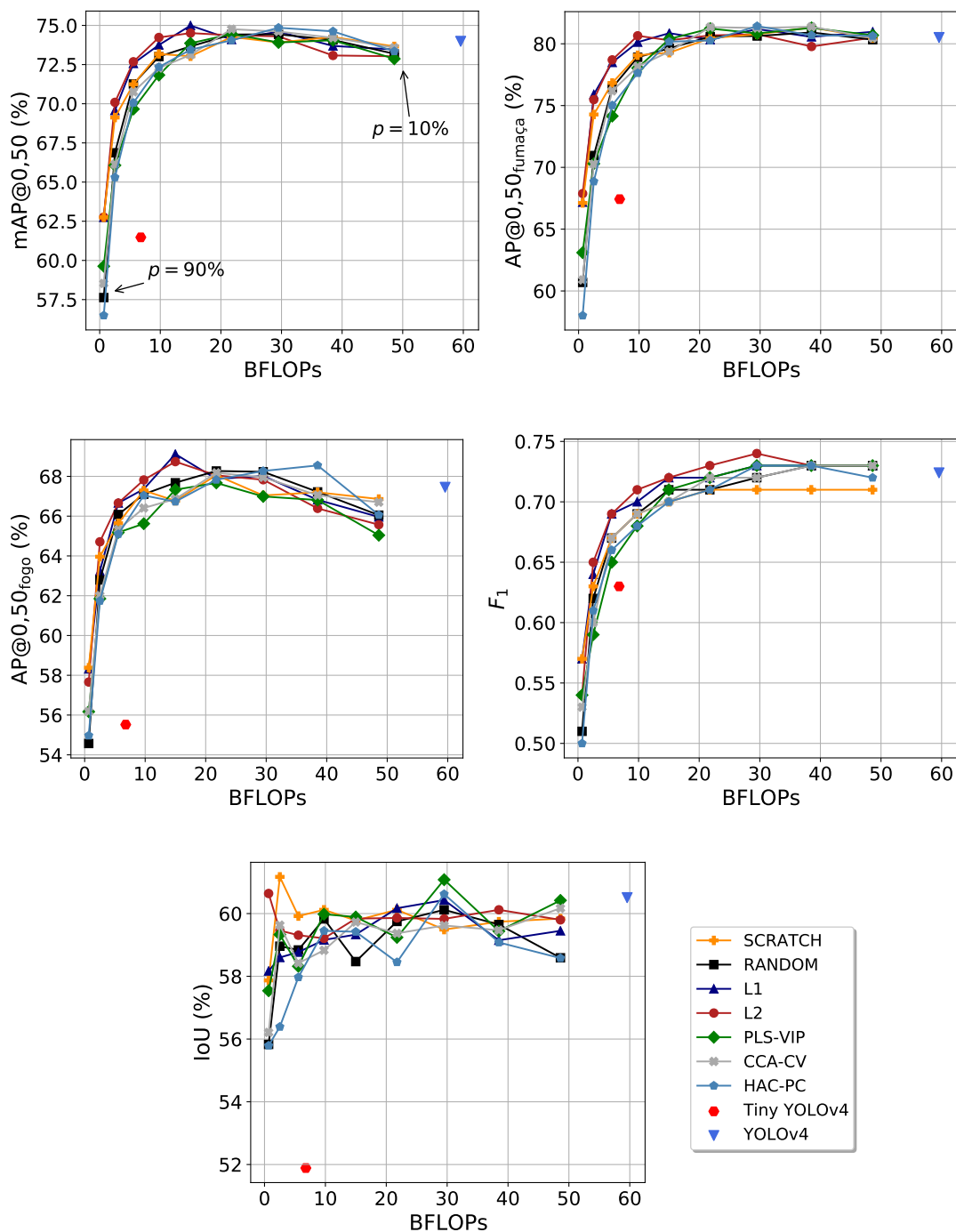


Figura 5.14: Relações de compromisso obtidas pelas técnicas de poda consideradas.

Tabela 5.5: Desempenho dos modelos podados pelas técnicas consideradas segundo o mAP@0,50 no conjunto de teste.

p	SCRATCH	RANDOM	L1	L2	PLS-VIP	CCA-CV	HAC-PC
10%	↓ 73,66%	↓ 73,19%	↓ 73,46%	↓ 73,03%	↓ 72,87%	↓ 73,58%	↓ 73,35%
20%	↑ 74,26%	↑ 74,07%	↓ 73,69%	↓ 73,08%	↑ 74,05%	↑ 74,21%	↑ 74,62%
30%	↓ 73,91%	↑ 74,43%	↑ 74,60%	↑ 74,30%	↓ 73,91%	↑ 74,61%	↑ 74,85%
40%	↑ 74,26%	↑ 74,42%	↑ 74,10%	↑ 74,37%	↑ 74,45%	↑ 74,76%	↑ 74,05%
50%	↓ 73,03%	↓ 73,66%	↑ 74,99%	↑ 74,51%	↓ 73,86%	↓ 73,12%	↓ 73,45%
60%	↓ 73,18%	↓ 73,01%	↓ 73,75%	↑ 74,23%	↓ 71,83%	↓ 72,28%	↓ 72,34%
70%	↓ 71,24%	↓ 71,26%	↓ 72,57%	↓ 72,68%	↓ 69,66%	↓ 70,76%	↓ 70,06%
80%	↓ 69,12%	↓ 66,87%	↓ 69,57%	↓ 70,09%	↓ 66,07%	↓ 66,14%	↓ 65,30%
90%	↓ 62,75%	↓ 57,63%	↓ 62,75%	↓ 62,77%	↓ 59,63%	↓ 58,53%	↓ 56,49%

Tabela 5.6: Desempenho dos modelos podados pelas técnicas consideradas segundo o AP@0,50 de fumaça no conjunto de teste.

p	SCRATCH	RANDOM	L1	L2	PLS-VIP	CCA-CV	HAC-PC
10%	↓ 80,47%	↓ 80,33%	↑ 80,97%	↓ 80,50%	↑ 80,69%	↓ 80,46%	↑ 80,64%
20%	↑ 81,32%	↑ 80,92%	↑ 80,56%	↓ 79,78%	↑ 81,29%	↑ 81,38%	↑ 80,69%
30%	↑ 80,77%	↑ 80,63%	↑ 81,19%	↑ 80,77%	↑ 80,84%	↑ 81,28%	↑ 81,43%
40%	↓ 80,45%	↑ 80,58%	↓ 80,33%	↑ 80,68%	↑ 81,23%	↑ 81,35%	↓ 80,27%
50%	↓ 79,28%	↓ 79,63%	↑ 80,86%	↓ 80,28%	↓ 80,39%	↓ 79,37%	↓ 80,17%
60%	↓ 79,07%	↓ 78,91%	↓ 80,13%	↑ 80,65%	↓ 78,04%	↓ 78,15%	↓ 77,63%
70%	↓ 76,83%	↓ 76,44%	↓ 78,49%	↓ 78,70%	↓ 74,15%	↓ 76,19%	↓ 75,02%
80%	↓ 74,28%	↓ 70,96%	↓ 75,90%	↓ 75,48%	↓ 70,30%	↓ 70,26%	↓ 68,86%
90%	↓ 67,12%	↓ 60,69%	↓ 67,18%	↓ 67,88%	↓ 63,10%	↓ 60,89%	↓ 58,01%

Tabela 5.7: Desempenho dos modelos podados pelas técnicas consideradas segundo o AP@0,50 de fogo no conjunto de teste.

p	SCRATCH	RANDOM	L1	L2	PLS-VIP	CCA-CV	HAC-PC
10%	↓ 66,86%	↓ 66,05%	↓ 65,96%	↓ 65,57%	↓ 65,04%	↓ 66,70%	↓ 66,06%
20%	↓ 67,20%	↓ 67,22%	↓ 66,82%	↓ 66,39%	↓ 66,81%	↓ 67,05%	↑ 68,55%
30%	↓ 67,04%	↑ 68,23%	↑ 68,02%	↑ 67,83%	↓ 66,99%	↑ 67,95%	↑ 68,27%
40%	↑ 68,06%	↑ 68,27%	↑ 67,87%	↑ 68,05%	↑ 67,67%	↑ 68,18%	↑ 67,83%
50%	↓ 66,77%	↑ 67,68%	↑ 69,12%	↑ 68,74%	↓ 67,33%	↓ 66,86%	↓ 66,73%
60%	↓ 67,29%	↓ 67,10%	↓ 67,36%	↑ 67,82%	↓ 65,62%	↓ 66,41%	↓ 67,04%
70%	↓ 65,64%	↓ 66,08%	↓ 66,66%	↓ 66,66%	↓ 65,18%	↓ 65,33%	↓ 65,10%
80%	↓ 63,96%	↓ 62,79%	↓ 63,24%	↓ 64,71%	↓ 61,85%	↓ 62,01%	↓ 61,74%
90%	↓ 58,38%	↓ 54,57%	↓ 58,32%	↓ 57,66%	↓ 56,17%	↓ 56,18%	↓ 54,97%

Tabela 5.8: Desempenho dos modelos podados pelas técnicas consideradas segundo a medida F_1 no conjunto de teste.

p	SCRATCH	RANDOM	L1	L2	PLS-VIP	CCA-CV	HAC-PC
10%	↓ 0,71	↑ 0,73	↑ 0,73	↑ 0,73	↑ 0,73	↑ 0,73	• 0,72
20%	↓ 0,71	↑ 0,73	↑ 0,73	↑ 0,73	↑ 0,73	↑ 0,73	↑ 0,73
30%	↓ 0,71	• 0,72	↑ 0,73	↑ 0,74	↑ 0,73	• 0,72	↑ 0,73
40%	↓ 0,71	↓ 0,71	• 0,72	↑ 0,73	• 0,72	• 0,72	↓ 0,71
50%	↓ 0,70	↓ 0,71	• 0,72	• 0,72	↓ 0,71	↓ 0,70	↓ 0,70
60%	↓ 0,69	↓ 0,69	↓ 0,70	↓ 0,71	↓ 0,68	↓ 0,69	↓ 0,68
70%	↓ 0,67	↓ 0,67	↓ 0,69	↓ 0,69	↓ 0,65	↓ 0,67	↓ 0,66
80%	↓ 0,63	↓ 0,62	↓ 0,64	↓ 0,65	↓ 0,59	↓ 0,60	↓ 0,61
90%	↓ 0,57	↓ 0,51	↓ 0,57	↓ 0,54	↓ 0,54	↓ 0,53	↓ 0,50

Tabela 5.9: Desempenho dos modelos podados pelas técnicas consideradas segundo a medida IoU no conjunto de teste.

p	SCRATCH	RANDOM	L1	L2	PLS-VIP	CCA-CV	HAC-PC
10%	↓ 59,84%	↓ 58,59%	↓ 59,45%	↓ 59,80%	↓ 60,42%	↓ 60,14%	↓ 58,58%
20%	↓ 59,74%	↓ 59,65%	↓ 59,15%	↓ 60,12%	↓ 59,44%	↓ 59,47%	↓ 59,08%
30%	↓ 59,49%	↓ 60,12%	↓ 60,43%	↓ 59,83%	↑ 61,08%	↓ 59,62%	↑ 60,62%
40%	↓ 60,12%	↓ 59,75%	↓ 60,17%	↓ 59,86%	↓ 59,24%	↓ 59,37%	↓ 58,45%
50%	↓ 59,78%	↓ 58,47%	↓ 59,33%	↓ 59,84%	↓ 59,89%	↓ 59,73%	↓ 59,42%
60%	↓ 60,12%	↓ 59,83%	↓ 59,16%	↓ 59,21%	↓ 59,98%	↓ 58,83%	↓ 59,45%
70%	↓ 59,92%	↓ 58,84%	↓ 58,77%	↓ 59,31%	↓ 58,32%	↓ 58,42%	↓ 57,97%
80%	↑ 61,17%	↓ 58,96%	↓ 58,60%	↓ 59,46%	↓ 59,33%	↓ 59,63%	↓ 56,39%
90%	↓ 57,87%	↓ 55,83%	↓ 58,17%	↑ 60,64%	↓ 57,54%	↓ 56,21%	↓ 55,79%

Diante das curvas apresentadas na Figura 5.14, é possível evidenciar, em qualquer métrica considerada, que não há uma grande diferença entre os desempenhos das redes otimizadas por diferentes técnicas de poda. Esses resultados são análogos aos encontrados no trabalho de Mittal et al. [2018] com a rede de detecção *Faster R-CNN* e a base de dados PASCAL VOC 2007 e indicam que a rede YOLOv4 no presente contexto também apresenta uma alta plasticidade. Em outras palavras, dada uma mesma taxa de poda, a rede de detecção de incêndios atinge níveis similares de recuperação de desempenho após o *fine-tuning* independentemente da técnica utilizada para seleção dos filtros convolucionais menos promissores. Além disso, é importante ressaltar que as curvas obtidas em todas as métricas dispõem de soluções dominadas, isto é, modelos que são superados por um ou mais modelos da mesma técnica em ambos os objetivos, como é o caso do modelo podado com $p = 50\%$ pela técnica L1, que domina todos os outros modelos da curva com $p \leq 40\%$. Sendo assim, como esses modelos não foram excluídos das soluções de compromisso, as curvas não podem ser tratadas como estimativas da fronteira Pareto-ótima.

Apesar da pequena diferença entre os métodos e embora cada modelo tenha se submetido ao processo de *fine-tuning* uma única vez em virtude do alto custo computacional envolvido, o que inviabiliza uma análise estatística dos resultados, algumas informações interessantes podem ser identificadas. À medida que a taxa de poda aumenta, por exemplo, maiores são as diferenças de desempenho obtidas entre as técnicas, especialmente para as métricas mAP@0,50, AP@0,50 (fogo e fumaça) e F_1 . Para valores de p maiores ou iguais a 50%, nota-se nas Tabelas 5.5 a 5.7 que, na maioria dos casos, ao menos uma das técnicas baseadas em normas superam todas as demais técnicas e que RANDOM e HAC-PC possuem tipicamente os piores desempenhos.

Outra particularidade a ser destacada da Figura 5.14 é o comportamento dos modelos quanto ao IoU. Ao contrário do que se observa nas demais métricas de avaliação consideradas, as curvas não apresentam uma relação de compromisso notável entre o desempenho e o custo computacional das redes para diferentes taxas de poda. Isso ocorre porque o IoU mensura apenas a qualidade das caixas delimitadoras previstas em relação às caixas delimitadoras reais e, portanto, não penaliza os valores calculados nos casos em que a rede deixa de detectar um ou mais objeto na cena (falsos negativos). Desse modo, altos valores de IoU não correspondem a um número elevado de detecções de fogo e fumaça, mas sim boas localizações das ocorrências de incêndio que foram identificadas corretamente.

No que se refere à métrica mAP@0,50, bem como as métricas AP@0,50 das classes fumaça e fogo, uma constatação intrigante é que mesmo não aproveitando os valores dos filtros remanescentes como inicialização para o reajuste da rede, a técnica SCRATCH ainda apresentou resultados competitivos. Contudo, esses resultados devem ser interpretados com uma certa ponderação. A métrica AP consiste na média das avaliações de precisão em todos os valores de revocação, considerando assim uma faixa de valores de limiar de confiança para cálculo. As redes de detecção treinadas, no entanto, operam no ambiente de produção com um limiar de confiança fixo, cujo melhor valor para o YOLOv4 foi definido anteriormente em $t_{\text{conf}} = 0,4$.

Uma comparação mais adequada pode ser realizada conforme a medida F_1 , que permite quantificar o comportamento da rede exclusivamente para o melhor limiar encontrado. Nesse caso, os modelos resultantes das podas com $p \leq 30\%$ por todas as técnicas são superiores aos modelos obtidos pela técnica SCRATCH, ou seja, apresentam um melhor *trade-off* entre verdadeiros positivos, falsos positivos e falsos negativos. Para taxas de poda maiores, isso deixa de ser verdade. A partir de uma taxa de poda de 40%, tem-se modelos bastante reduzidos em complexidade, tal que ao menos 63,80% dos parâmetros já foram removidos (Tabela 5.4). Assim, uma vez que o número de iterações para *fine-tuning* foi fixado em 30.000 para qualquer taxa de poda, os filtros

restantes são capazes de convergir para um ótimo local com qualidade similar aos ótimos locais encontrados pelas outras técnicas mesmo sendo inicializados aleatoriamente.

Sob essa perspectiva, um experimento adicional foi realizado com o intuito de analisar os desempenhos dos modelos podados com $p = 80\%$ por cada uma das técnicas em função do número de iterações para *fine-tuning*. Foram avaliados números de iterações uniformemente espaçados no intervalo de 5.000 a 30.000, com passos de 5.000 iterações, tal que em cada experimento a taxa de aprendizado foi ajustada com um fator de redução $\gamma = 0,1$ nos marcos de 80% e 90% das iterações totais. Por exemplo, considerando um *fine-tuning* de 15.000 iterações, a taxa de aprendizado sofrerá uma redução de 90% nas iterações 12.000 e 13.500. Os demais hiperparâmetros foram mantidos conforme especificado na Seção 5.2.

A Figura 5.15 mostra que a diferença de desempenho das redes resultantes diminui com o aumento do número de iterações utilizados no processo de *fine-tuning*. A maior diferença em 5.000 iterações foi de 10,56 p.p. de $mAP@0,50$ e 0,15 de F_1 entre as técnicas L2 e HAC-PC. Em 30.000 iterações, a diferença entre essas mesmas técnicas foi de 4,79 p.p. de $mAP@0,50$ e 0,04 de F_1 . Essa tendência de convergência pode ser observada ainda a partir dos segmentos de reta presentes entre pontos consecutivos da curva, que diminuem de inclinação com o aumento das iterações. Para as técnicas SCRATCH e PLS-VIP, a inclinação entre os pontos 25.000 e 30.000 ficaram negativas na medida F_1 , sugerindo ainda que ajustes excessivos nos parâmetros podem depreciar o potencial de generalização da rede.

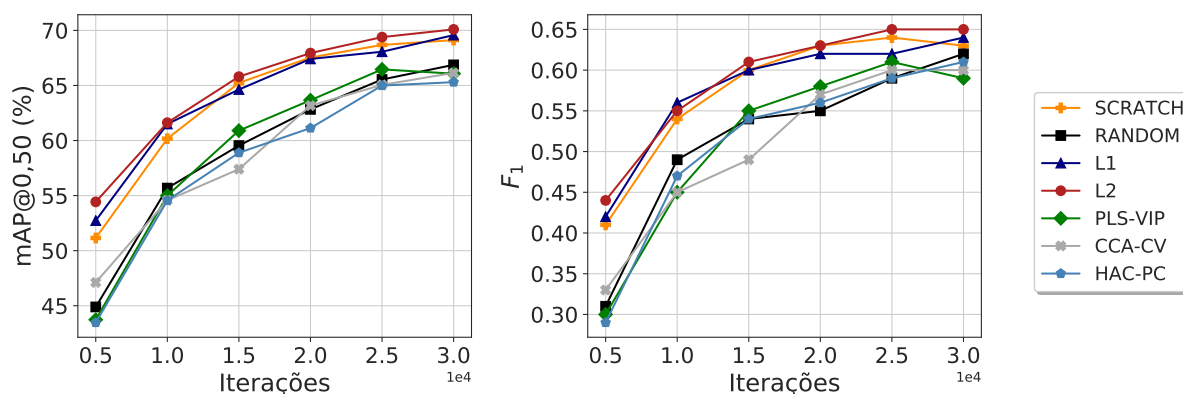


Figura 5.15: Desempenho dos modelos podados com $p = 80\%$ por cada uma das técnicas de poda, segundo as métricas $mAP@0,50$ (à esquerda) e F_1 (à direita) no conjunto de teste, em função do número de iterações.

Por fim, constata-se que é possível preservar ou até mesmo superar o desempenho das redes de detecção de incêndios apresentadas na Tabela 5.1 a um custo computacio-

nal bastante reduzido. Ao podar 40% dos filtros convolucionais de cada camada da rede YOLOv4 com as técnicas baseadas em múltiplas projeções (PLS-VIP e CCA-CV), por exemplo, tem-se resultados melhores no mAP, $AP_{\text{fumaça}}$ e AP_{fogo} ($t_{\text{IoU}} = 0, 50$), resultados equivalentes na medida F_1 e resultados ligeiramente inferiores no IoU, isto é, cerca de apenas 1,2 p.p. Enquanto que essas redes otimizadas agora detectam com 37,812 BFLOPs a menos por propagação direta e requerem 163,29MB a menos de memória para armazenamento quando comparadas à rede YOLOv4 original.

Em cenários onde são requisitadas redes ainda mais leves, tem-se que todos os modelos podados com $p = 70\%$ superam significativamente a rede Tiny YOLOv4 nas cinco métricas de avaliação investigadas, bem como reduzem o custo computacional de detecção e o consumo de memória em 18,1% e 0,8%, respectivamente. No caso de melhor desempenho com $p = 70\%$, onde a rede é podada com a técnica L2, as melhorias são de 11,21 p.p. no mAP@0,50, 11,27 p.p. no $AP@0,50_{\text{fumaça}}$, 11,14 p.p. no $AP@0,50_{\text{fogo}}$, 0,06 na medida F_1 e 7,42 p.p. no IoU, corroborando novamente os benefícios de se remover os filtros convolucionais redundantes ou irrelevantes da rede.

5.5 Estudo de caso

A fim de analisar a velocidade dos modelos de detecção de incêndios otimizados em um dispositivo com orçamento computacional limitado, um estudo de caso foi realizado em um microcomputador de placa única (*single board computer*) Raspberry Pi 4 modelo B [Raspberry Pi Foundation, 2021], cujas especificações técnicas são:

- **Processador:** Broadcom BCM2711, Quad Core Cortex-A72 (ARM v8) 64-bit SoC @ 1,5GHz.
- **Memória RAM:** 4GB LPDDR4-3200 SDRAM.
- **Gráficos:** OpenGL ES 3.0, com codificadores H.265 (4Kp60), H264 (1080p60, 1080p30).
- **Saídas de vídeo:** 2 × microHDMI (com suporte até 4Kp60).
- **Conectividade:** WiFi 2.4 GHz e 5.0 GHz IEEE 802.11ac, Bluetooth 5.0, BLE e Gigabit Ethernet.
- **Portas USB:** 2 portas USB 3.0 e 2 portas USB 2.0.
- **MicroSD:** Cartão de Memória SanDisk Ultra 32GB.

Os experimentos conduzidos visaram investigar dois fatores principais: (i) tempo de inicialização do modelo, isto é, tempo necessário para carregar todos os parâmetros

da rede e (ii) tempo de detecção, isto é, tempo decorrido para identificar as ocorrências de fogo e fumaça em uma imagem de dimensão 416×416 no espaço de cor RGB.

Em concordância com a Tabela 5.4, a relação do custo computacional dos modelos com a intensidade de poda empregada é igual em todas as técnicas. Assim, os tempos de inicialização e detecção dos modelos obtidos com diferentes técnicas para uma mesma taxa de poda serão praticamente idênticos. Perante o exposto, os experimentos são mostrados apenas para a técnica L1. Cada um dos modelos foi executado 35 vezes. De modo a não depreciar o desempenho da placa em função de um superaquecimento ($\geq 80^\circ\text{C}$) durante os experimentos, um cooler com duas ventoinhas foi integrado ao sistema embarcado descrito.

A partir da Figura 5.16, constata-se que à medida que a poda da arquitetura se torna mais intensa, os tempos médios de inicialização e detecção diminuem. No caso particular do tempo de detecção, onde as amplitudes interquartis das caixas apresentaram uma menor variabilidade, essa observação é ainda mais fidedigna, visto que os intervalos de tempo obtidos com os diferentes modelos são disjuntos, isto é, não apresentam interseção. Quando as comparações são feitas entre os casos extremos de poda efetuados, isto é, a rede podada com $p = 10\%$ e a rede podada com $p = 90\%$, nota-se também que a diferença entre os tempos médios de inicialização é bastante pequena em relação à diferença entre os tempos médios de detecção. Esse fato corrobora ainda mais a pertinência da poda de filtros como alternativa para aumentar a velocidade de detecção das redes convolucionais, já que a inicialização de parâmetros é feita apenas uma única vez e a detecção, em contrapartida, é feita para cada quadro de vídeo.

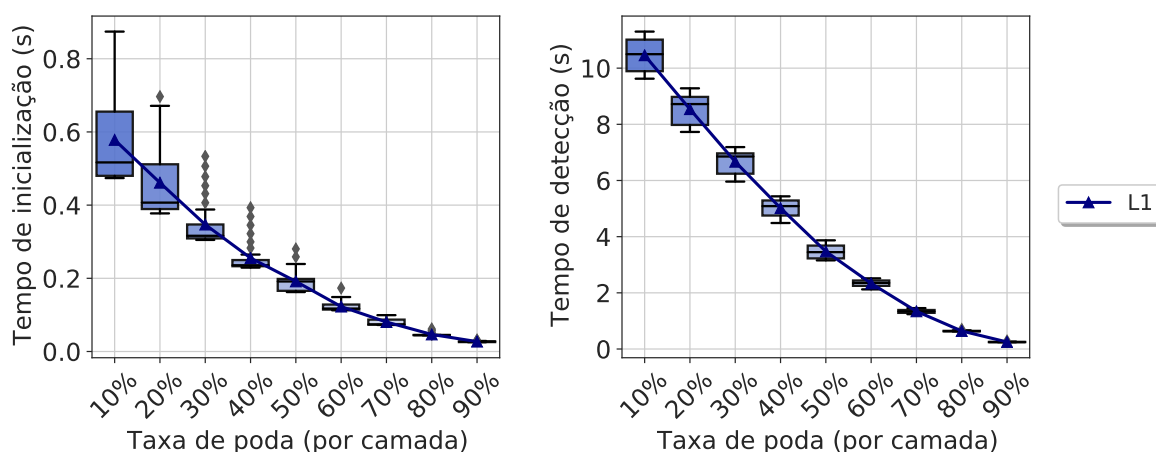


Figura 5.16: Relação do tempo de inicialização (à esquerda) e do tempo de detecção (à direita) no Raspberry Pi 4 com a intensidade de poda empregada. Os diagramas de caixa compreendem as distribuições dos tempos e os gráficos de linha representam os tempos médios.

Embora as reduções nos tempos sejam significativas à medida que a taxa de poda aumenta, esse comportamento constatado é previsto, já que a poda implica diretamente na redução de parâmetros do modelo, bem como na diminuição do número de operações de ponto flutuante realizadas na etapa de propagação direta da rede. O ideal é que se tenha o máximo de redução desses tempos com a menor depreciação possível no desempenho das detecções de fogo e fumaça. Em prol de examinar esse compromisso, foram apresentadas comparações dos modelos obtidos pela técnica L1 com as performances médias, também em 35 execuções no Raspberry, das arquiteturas Tiny YOLOv4 e YOLOv4.

Observa-se na Figura 5.17 que os modelos otimizados com as taxas de poda $p = \{30\%, 40\%, 50\%\}$ dominam o YOLOv4 ($p = 0\%$) nos dois objetivos de cada curva. No melhor cenário dentre esses três modelos, ou seja, com $p = 50\%$, é possível obter um detector de incêndios com 1 p.p. de mAP@0,50 a mais no conjunto de teste e que demanda, em média, menos de $1/5$ do tempo de inicialização e menos de $1/4$ do tempo de detecção quando comparado à rede YOLOv4 sem qualquer tipo de otimização. Supondo uma máquina com um processamento ainda mais proibitivo, uma possível alternativa seria o modelo podado com $p = 70\%$, cujos tempos médios de inicialização e detecção são equivalentes ao da rede Tiny YOLOv4, porém com aproximadamente 11,1 p.p. de mAP@0,50 a mais no conjunto de teste.

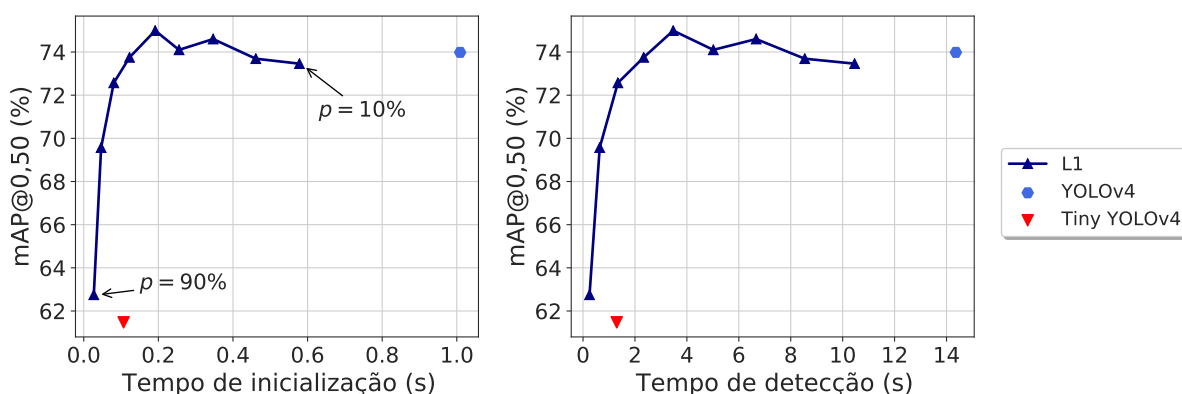


Figura 5.17: Relação do desempenho dos modelos segundo a métrica mAP@0,50 no conjunto de teste com o tempo médio de inicialização (à esquerda) e com o tempo médio de detecção (à direita) no Raspberry Pi 4.

Perante o exposto, constata-se uma possível oportunidade de se monitorar grandes áreas verdes com eficiência a partir de múltiplas câmeras. Cada câmera instalada seria integrada a um microcomputador responsável pela execução do detector de incêndios baseado em rede convolucional profunda. A infraestrutura em pauta, entretanto, não demandaria servidores providos de unidades de processamento gráfico e nem de

computação em nuvem, sendo ambas alternativas muito mais caras do que algumas dezenas de microcomputadores de placa única.

5.6 Conclusões do capítulo

Neste capítulo foram apresentados os experimentos conduzidos para construção de um detector de incêndios robusto e computacionalmente factível para dispositivos de baixo poder computacional. Os resultados demonstraram que treinar redes mais complexas e depois otimizá-las através da remoção dos filtros convolucionais menos importantes é uma estratégia muito eficiente e pode resultar em redes melhores e mais rápidas do que redes menores treinadas do zero (*from scratch*). No que tange as comparações entre as redes resultantes de diferentes técnicas em podas de uma única iteração, foi evidenciado que as técnicas baseadas em normas se mostraram mais efetivas para remoções maiores ou iguais a 50% dos filtros em cada camada. Além disso, não foram verificadas diferenças muito grandes para taxas menores ou iguais a 30%. No entanto, como as técnicas baseadas em normas são menos dispendiosas no processo de seleção dos filtros a serem eliminados, isto é, não precisam de processar dados para ranquear os filtros por importância (*data-agnostic*), elas também são preferíveis nesse caso.

Capítulo 6

Conclusões

Dentre as abordagens modernas desenvolvidas para detecção automática de fogo e fumaça, principais indicadores visuais do incêndio, as mais notáveis são tipicamente fundamentadas em redes convolucionais profundas. Diferentemente dos métodos clássicos de extração de características, essa topologia inspirada na percepção visual é capaz de extrair automaticamente características de alta relevância da cena. Contudo, as CNNs dispõem de dois desafios principais: exigem um grande número de imagens para treinamento de seus parâmetros e também um elevado poder computacional para operarem com êxito, o que torna as propostas atuais para identificação de incêndios muito onerosas em larga escala.

Em virtude da demanda por imagens e da escassez de conjuntos de dados rotulados na literatura para a tarefa de detecção de incêndios, o presente trabalho propôs a D-Fire, uma base com mais de 21.000 imagens contendo eventos reais e sintéticos de fogo e fumaça, bem como as anotações das coordenadas das caixas delimitadoras de todos esses eventos. Mediante o conjunto de dados, um detector de incêndios foi construído a partir do treinamento da quarta versão da rede *You Only Look Once* (YOLOv4). Os resultados obtidos evidenciaram uma notável robustez do detector a cenários com grau de incerteza elevado e uma grande capacidade de generalização às diversas cores, formas e posições assumidas por fogo e fumaça.

No que se refere ao dilema do custo computacional proibitivo para executar o detector em dispositivos de processamento limitado, principalmente em tempo hábil para emissão de alerta às entidades responsáveis pela supressão dos incêndios, esta pesquisa buscou otimizar eficientemente a arquitetura da rede, de modo que o desempenho já certificado não fosse comprometido. Uma vez que o YOLOv4 possui cerca de 64 milhões de parâmetros, tal que a maioria provêm dos 33.917 filtros convolucionais dispostos em sua arquitetura, nada mais coerente que remover os filtros menos importantes de cada camada para diminuir a complexidade da rede.

Para tal fim, foram investigados os efeitos de diferentes técnicas de poda de filtros no detector de incêndios desenvolvido. Embora não se tenha observado superioridade

de uma técnica em particular em relação às demais em todas as taxas de poda consideradas, no melhor caso de desempenho, onde foi empregada a técnica L1 com poda de 50% dos filtros do YOLOv4 em cada camada, obteve-se reduções de 74,96% dos parâmetros da rede, 74,93% de memória e 74,84% de custo computacional e um aumento de 1,37% de mAP@0,50. A maior taxa de poda que resulta numa rede cujo desempenho de detecção mais se aproxima da rede não podada é $p = 60\%$ com a técnica L2, porém com cerca de 83,88% a menos de parâmetros, 83,86% a menos de memória e 83,60% a menos de BFLOPs por propagação direta. Em dispositivos ainda mais restritos computacionalmente, uma alternativa à rede Tiny YOLOv4 foi apresentada ao se podar 70% dos filtros da rede YOLOv4 com a técnica L2, tal que a rede resultante obteve custo computacional ligeiramente menor e um aumento de 18,24% de mAP@0,50.

6.1 Trabalhos futuros

Essa pesquisa apresentou resultados bastante interessantes para o contexto de monitoramento ambiental em larga escala, instigando um cenário promissor de detecção de incêndios com processamento local distribuído de baixo custo. No entanto, muitas novas possibilidades de investigação e de melhoria podem ser apontadas a partir do que foi feito. Algumas delas são:

- **Contexto temporal:** ainda que o aprendizado da rede tenha sido bem-sucedido em uma variedade de cenários, muitos objetos comuns apresentam uma alta similaridade visual com as classes fogo e fumaça, principalmente em determinadas condições de iluminação. Por exemplo, pequenos focos de fogo podem ser confundidos com luzes de postes públicos e faróis automotivos acesos em ambientes noturnos vistos a longa distância. Por consequência, o número de falsos positivos retornados pelo detector pode crescer consideravelmente, o que não é ideal para aplicações que mobilizam esforços humanos para contenção dos incêndios. Assim, uma possibilidade pertinente para contornar esse eventual problema é considerar o comportamento do fogo e da fumaça ao longo do tempo, já que ambos são bastante característicos. Essa análise temporal pode ser integrada ao detector proposto, por exemplo, como um pós-processamento das caixas delimitadoras preditas, de modo que a ocorrência seja certificada antes da emissão do alerta de incêndio.
- **Detecção de focos de incêndio pequenos:** durante a análise de desempenho da rede YOLOv4 feita neste trabalho, uma eminente discrepância de desempenho

entre as classes foi observada, tal que se obteve, em média, $AP@0,50 = 80,51\%$ na detecção de fumaça e $AP@0,50 = 67,45\%$ na detecção de fogo. Conforme mencionado anteriormente, em detrimento de uma detecção rápida, a arquitetura do YOLO, desde sua primeira versão, apresenta algumas limitações na detecção de objetos pequenos [Redmon et al., 2016]. Dado que o propósito de um detector de incêndios é sempre identificar os incêndios em seus estágios iniciais, essa particularidade do YOLO vai em direção oposta ao desejado nessas circunstâncias. Em busca de melhores resultados, outros detectores de objetos baseados em aprendizado profundo podem ser treinados na base de dados D-Fire para a tarefa de detecção de incêndios, como a EfficientDet [Tan et al., 2020] e o *You Only Learn One Representation* (YOLOR) [Wang et al., 2021].

- **Poda iterativa:** uma das premissas assumidas na metodologia deste estudo devido ao elevado custo computacional dos experimentos foi a investigação majoritária da poda direta, onde as etapas de remoção de filtros menos promissores e *fine-tuning* são realizadas apenas uma única vez. Contudo, para uma mesma taxa de poda final, empregar o método PLS-VIP iterativamente em redes convolucionais de classificação é bem mais eficaz do que empregá-lo diretamente [Jordao et al., 2020b]. Sob esse princípio, uma potencial direção de pesquisa é investigar a poda iterativa no presente problema, a fim de se observar melhores resultados, especialmente para podas mais intensas.
- **Poda de camadas convolucionais:** além da poda de filtros convolucionais, outra vertente de poda existente é a remoção das camadas convolucionais menos relevantes para o desempenho da rede. Alguns trabalhos na literatura sugerem que a poda de camadas pode apresentar resultados superiores aos da poda de filtros, tanto em termos de eficiência computacional quanto de performance preditiva [Veit et al., 2016; Veit e Belongie, 2018; Fan et al., 2020]. Diante disso, possíveis análises experimentais considerando poda de camadas convolucionais podem resultar em redes de detecção de incêndios ainda mais rápidas e robustas do que as apresentadas nesse trabalho.
- **Técnicas de poda baseadas em múltiplas projeções:** em um problema de classificação, cada imagem está associada a um único rótulo de classe. Já em um problema de detecção, cada imagem pode estar associada a um número variável de rótulos (múltiplos objetos), tal que rótulos de uma mesma imagem não representam obrigatoriamente a mesma classe (objetos distintos). A fim de adequar a representação da variável de saída originalmente proposta por Jordao

et al. [2020b] em tarefas de classificação para o presente problema de detecção, proposições bastante rígidas foram assumidas. No que diz respeito ao CCA-CV, mesmo sendo completamente dissimilares visualmente, as classes fogo e fumaça foram unificadas como uma única classe incêndio. No PLS-VIP, por sua vez, a classe de maior incidência na imagem foi assumida como rótulo, o que desconsidera a presença de múltiplos objetos distintos em uma cena e, portanto, prejudica a verdadeira relação dos filtros convolucionais com os rótulos em um espaço latente de baixa dimensão. Para atenuar o problema desse segundo caso, uma alternativa possivelmente promissora é modelar a saída através da suavização de rótulo (*class label smoothing*), que permite relaxar a confiança do modelo no rótulo verdadeiro ao definir cada rótulo como uma probabilidade de classe e não apenas como um valor binário [Szegedy et al., 2016].

Referências bibliográficas

- Aidouni, M. E. (2019). Evaluating Object Detection Models: Guide to Performance Metrics. <https://git.io/JIKFq>. Acesso em 11 de Dezembro de 2020.
- Anderson, L. O.; Burton, C.; dos Reis, J. B. C.; Pessôa, A. C. M.; Selaya, G.; Bett, P.; Jones, C.; Williams, K.; Taylor, I.; Wiltshire, A. e Aragão, L. (2020). Identification of priority areas for reducing the likelihood of burning and forest fires in South America August to October 2020. Relatório técnico, Climate Science for Service Partnership (CSSP) Brazil, São José dos Campos, Brazil.
- Angayarkkani, K. e Radhakrishnan, N. (2010). An Intelligent System For Effective Forest Fire Detection Using Spatial Data. *arXiv preprint arXiv:1002.2199*. ISSN 03604012.
- Anwar, S.; Hwang, K. e Sung, W. (2017). Structured Pruning of Deep Convolutional Neural Networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1--18.
- AsusTek Computer Inc. (2021). Tinker Board. <https://tinker-board.asus.com/product/tinker-board.html>. Acesso em 23 de Agosto de 2021.
- Barron, J. L.; Fleet, D. J. e Beauchemin, S. S. (1994). Performance of Optical Flow Techniques. *International Journal of Computer Vision*, 12(1):43--77.
- Batista, L. S. (2020). Notas de aula da disciplina Teoria da Decisão (ELE088): Otimização Restrita. www.ppgge.ufmg.br/~lusoba. Acesso em 19 de Novembro de 2020.
- Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1--127. ISSN 13448994.
- Blumer, A.; Ehrenfeucht, A.; Haussler, D. e Warmuth, M. K. (1987). Occam's Razor. *Information Processing Letters*, 24(6):377--380.
- Bochkovskiy, A.; Wang, C.-Y. e Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*.

- Borenstein, E.; Sharon, E. e Ullman, S. (2004). Combining Top-Down and Bottom-Up Segmentation. Em *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 46--46. IEEE.
- Bottou, L. (1998). Online Learning and Stochastic Approximations. *Online Learning in Neural Networks*.
- Boult, T. E.; Micheals, R. J.; Gao, X. e Eckmann, M. (2001). Into the Woods: Visual Surveillance of Noncooperative and Camouflaged Targets in Complex Outdoor Settings. *Proceedings of the IEEE*, 89(10):1382--1402. ISSN 00189219.
- Braga, A. P. (2000). *Redes Neurais Artificiais: Teoria e Aplicações*. Livros Técnicos e Científicos (LTC).
- Buciluă, C.; Caruana, R. e Niculescu-Mizil, A. (2006). Model Compression. Em *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 535--541.
- Buckland, M. e Gey, F. (1994). The Relationship between Recall and Precision. *Journal of the American Society for Information Science*, 45(1):12--19.
- Cauchy, A. (1847). Méthode Générale Pour la Résolution des Systemes D'équations Simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536--538.
- Celik, T. (2010). Fast and Efficient Method for Fire Detection using Image Processing. *ETRI Journal*, 32(6):881--890. ISSN 12256463.
- Celik, T. e Demirel, H. (2009). Fire Detection in Video Sequences using a Generic Color Model. *Fire Safety Journal*, 44(2):147--158. ISSN 03797112.
- Celik, T.; Demirel, H.; Ozkaramanli, H. e Uyguroglu, M. (2007). Fire Detection using Statistical Color Model in Video Sequences. *Journal of Visual Communication and Image Representation*, 18(2):176--185. ISSN 10473203.
- Center for Wildfire Research (CWR) (2010). Wildfire Observers and Smoke Recognition Homepage. <http://wildfire.fesb.hr/>. Acesso em 3 de Agosto de 2021.
- Çetin, A. E.; Dimitropoulos, K.; Gouverneur, B.; Grammalidis, N.; Günay, O.; Habiboğlu, Y. H.; Töreyn, B. U. e Verstockt, S. (2013). Video Fire Detection: Review. *Digital Signal Processing: A Review Journal*, 23(6):1827--1843. ISSN 10512004.

- Chen, G.; Choi, W.; Yu, X.; Han, T. e Chandraker, M. (2017). Learning Efficient Object Detection Models with Knowledge Distillation. *Advances in Neural Information Processing Systems*, 30.
- Chen, T.-H.; Wu, P.-H. e Chiou, Y.-C. (2004). An Early Fire-Detection Method Based on Image Processing. Em *International Conference on Image Processing (ICIP)*, volume 3, pp. 1707--1710.
- Chen, T.-H.; Yin, Y.-H.; Huang, S.-F. e Ye, Y.-T. (2006). The Smoke Detection for Early Fire-Alarming System Base on Video Processing. Em *International Conference on Intelligent Information Hiding and Multimedia*, pp. 427--430. IEEE.
- Cheng, C.; Sun, F. e Zhou, X. (2011). One Fire Detection Method using Neural Networks. *Tsinghua Science and Technology*, 16(1):31--35. ISSN 10070214.
- Cheng, X.; Wu, J.; Yuan, X. e Zhou, H. (1999). Principles for a Video Fire Detection System. *Fire Safety Journal*, 33(1):57--69. ISSN 03797112.
- Chino, D. Y.; Avalhais, L. P.; Rodrigues, J. F. e Traina, A. J. (2015). BoWFire: Detection of Fire in Still Images by Integrating Pixel Color and Texture Analysis. Em *28th SIBGRAPI conference on graphics, patterns and images*, pp. 95--102. IEEE.
- Chong, E. K. e Zak, S. H. (2004). *An Introduction to Optimization*. John Wiley & Sons, 4 edição. ISBN 9781118279014.
- Chunyu, Y.; Jun, F.; Jinjun, W. e Yongming, Z. (2010). Video Fire Smoke Detection using Motion and Color Features. *Fire Technology*, 46(3):651--663. ISSN 00152684.
- Ciresan, D. C.; Meier, U.; Masci, J.; Gambardella, L. M. e Schmidhuber, J. (2011). Flexible, High Performance Convolutional Neural Networks for Image Classification. Em *Twenty-second International Joint Conference on Artificial Intelligence*.
- Clevert, D. A.; Unterthiner, T. e Hochreiter, S. (2016). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1--14.
- Collins, R. T.; Lipton, A. J.; Kanade, T.; Fujiyoshi, H.; Duggins, D.; Tsin, Y.; Tolliver, D.; Enomoto, N.; Hasegawa, O.; Burt, P. e Wixson, L. (2000). A System for Video Surveillance and Monitoring. *VSAM final report*, pp. 1--68.
- Data Science Academy (2019). Deep Learning Book. <http://deeplearningbook.com.br/>. Acesso em 19 de Novembro de 2020.

- Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K. e Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. Em *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248--255. IEEE.
- Dhingra, A. (2017). Model Complexity-Accuracy Trade-off for a Convolutional Neural Network. *arXiv preprint arXiv:1705.03338*.
- Dugas, C.; Bengio, Y.; Bélisle, F.; Nadeau, C. e Garcia, R. (2001). Incorporating Second-Order Functional Knowledge for Better Option Pricing. *Advances in Neural Information Processing Systems*. ISSN 10495258.
- Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J. e Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International journal of computer vision*, 88(2):303--338.
- Fan, A.; Grave, E. e Joulin, A. (2020). Reducing Transformer Depth on Demand with Structured Dropout. Em *International Conference on Learning Representations*.
- Fernández, A.; García, S.; Galar, M.; Prati, R. C.; Krawczyk, B. e Herrera, F. (2018). *Learning from Imbalanced Data Sets*. Springer.
- Friedman, J. H. (1997). On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55--77.
- Friendlander, S. K. (2000). *Smoke, Dust, and Haze: Fundamentals of Aerosol Dynamics*. Oxford University Press, New York, USA, 2nd edição.
- Frizzi, S.; Kaabi, R.; Bouchouicha, M.; Ginoux, J. M.; Moreau, E. e Fnaiech, F. (2016). Convolutional Neural Network for Video Fire and Smoke Detection. Em *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pp. 877--882. IEEE.
- Fukushima, K. (1980). Neocognition: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological cybernetics*, 36(4):193--202. ISSN 03401200.
- Gaia (2018). D-Fire: An image dataset for fire detection. <https://github.com/gaiasd/DFireDataset>. Acesso em 1 de Fevereiro de 2021.
- Geman, S.; Bienenstock, E. e Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1--58.

- Ghiasi, G.; Lin, T.-Y. e Le, Q. V. (2018). Dropblock: A Regularization Method for Convolutional Networks. *arXiv preprint arXiv:1810.12890*.
- Ghosh, S.; Srinivasa, S. K.; Amon, P.; Hutter, A. e Kaup, A. (2019). Deep Network Pruning for Object Detection. Em *IEEE International Conference on Image Processing (ICIP)*, pp. 3915--3919. IEEE.
- Girshick, R. (2015). Fast r-cnn. Em *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440--1448.
- Glorot, X. e Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feed-forward Neural Networks. Em *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249--256.
- Goodfellow, I.; Bengio, Y. e Courville, A. (2016). *Deep Learning*. MIT Press.
- Gotmare, A.; Keskar, N. S.; Xiong, C. e Socher, R. (2018). A Closer Look at Deep Learning Heuristics: Learning Rate Restarts, Warmup and Distillation. *arXiv preprint arXiv:1810.13243*.
- Govil, K.; Welch, M. L.; Ball, J. T. e Pennypacker, C. R. (2020). Preliminary Results from a Wildfire Detection System Using Deep Learning on Remote Camera Images. *Remote Sensing*, 12(1):166.
- Goyal, M.; Rajpura, P.; Bojinov, H. e Hegde, R. (2017a). Dataset Augmentation with Synthetic Images Improves Semantic Segmentation. Em *National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics*, pp. 348--359. Springer.
- Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y. e He, K. (2017b). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv preprint arXiv:1706.02677*.
- Guillemant, P. e Vicente, J. (2001). Real-Time Identification of Smoke Images by Clustering Motions on a Fractal Curve with a Temporal Embedding Method. *Optical Engineering*, 40(4):554--564.
- Guyon, I. e Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3(Mar):1157--1182.
- Han, S.; Pool, J.; Tran, J. e Dally, W. (2015). Learning Both Weights and Connections for Efficient Neural Network. *Advances in Neural Information Processing Systems*, 28:1135--1143.

- Hao, S.; Zhou, Y. e Guo, Y. (2020). A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing*, 406:302--321.
- He, K.; Zhang, X.; Ren, S. e Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904--1916.
- He, K.; Zhang, X.; Ren, S. e Sun, J. (2016). Deep Residual Learning for Image Recognition. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770--778.
- He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J. e Li, M. (2019). Bag of Tricks for Image Classification with Convolutional Neural Networks. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 558--567.
- He, Y.; Kang, G.; Dong, X.; Fu, Y. e Yang, Y. (2018). Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. *arXiv preprint arXiv:1808.06866*.
- He, Y.; Zhang, X. e Sun, J. (2017). Channel Pruning for Accelerating Very Deep Neural Networks. Em *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389--1397.
- Hinton, G.; Vinyals, O. e Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.
- Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J. et al. (2001). Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies.
- Hochreiter, S. e Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735--1780.
- Hohberg, S. P. (2015). *Wildfire Smoke Detection using Convolutional Neural Networks*. Tese de doutorado, Freie Universität Berlin.
- Hoiem, D.; Chodpathumwan, Y. e Dai, Q. (2012). Diagnosing Error in Object Detectors. Em *European Conference on Computer Vision*, pp. 340--353. Springer.
- Hossin, M. e Sulaiman, M. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):1.

- Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M. e Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*.
- Hu, H.; Peng, R.; Tai, Y.-W. e Tang, C.-K. (2016). Network Trimming: A Data-Driven Neuron Pruning Approach Towards Efficient Deep Architectures. *arXiv preprint arXiv:1607.03250*.
- Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S. et al. (2017). Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7310--7311.
- Hui, J. (2018). mAP (mean Average Precision) for Object Detection. <https://bit.ly/3aGCZxW>. Acesso em 11 de Dezembro de 2020.
- INPE (2020). Programa Queimadas: Dados Atuais do Satélite de Referência AQUA Tarde. <http://queimadas.dgi.inpe.br/queimadas/portal-static/situacao-atual/>. Acesso em 24 de Dezembro de 2020.
- Ioffe, S. e Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167*.
- Jaitly, N. e Hinton, G. E. (2013). Vocal Tract Length Perturbation (VTLP) Improves Speech Recognition. Em *Proceedings ICML Workshop on Deep Learning for Audio, Speech and Language*, volume 117.
- Jiang, X.; Hadid, A.; Pang, Y.; Granger, E. e Feng, X. (2019). *Deep Learning in Object Detection and Recognition*. Springer.
- Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z. e Qu, R. (2019). A Survey of Deep Learning-Based Object Detection. *IEEE Access*, 7:128837--128868.
- Jiaxin Gu (2017). BBox Label Tool Multiclass. <https://git.io/JBjTE>. Acesso em 8 de Janeiro de 2018.
- Jocher, G.; Kwon, Y.; Veitch-Michaelis, J.; Bianconi, G.; Baltacı, F.; Suess, D.; Xinyu, W. e et al. (2020). ultralytics/yolov3: 43.1mAP@0.5:0.95 on COCO2014. <https://doi.org/10.5281/zenodo.3785397>. Versão v7 publicada em Maio de 2020 (10.5281/zenodo.3785397).

- Joglar, F.; Risk, T.; Mowrer, F. e Protection, F. (2005). A Probabilistic Model for Fire Detection with Applications. *Fire Technology*, 41:151--172.
- Jones, G. A.; Paragios, N. e Regazzoni, C. S. (2012). *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*. Springer Science & Business Media.
- Jordao, A.; Lie, M. e Schwartz, W. R. (2020a). Discriminative Layer Pruning for Convolutional Neural Networks. *IEEE Journal of Selected Topics in Signal Processing*.
- Jordao, A.; Yamada, F. e Schwartz, W. R. (2020b). Deep Network Compression based on Partial Least Squares. *Neurocomputing*, 406:234 – 243. ISSN 0925-2312.
- Karnik, N. N. e Mendel, J. M. (1998). Introduction to Type-2 Fuzzy Logic Systems. Em *International Conference on Fuzzy Systems Proceedings*, volume 2, pp. 915--920. IEEE.
- Klambauer, G.; Unterthiner, T.; Mayr, A. e Hochreiter, S. (2017). Self-Normalizing Neural Networks. *Advances in Neural Information Processing Systems*, 2017-December:972--981. ISSN 10495258.
- Koehn, P. (2009). *Statistical Machine Translation*. Cambridge University Press.
- Kopilovic, I.; Vagvolgyi, B. e Sziranyi, T. (2000). Application of Panoramic Annular Lens for Motion Analysis Tasks: Surveillance and Smoke Detection. Em *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 4, pp. 714--717. IEEE.
- Krizhevsky, A. e Hinton, G. (2010). Convolutional Deep Belief Networks on CIFAR-10. *Unpublished manuscript*, pp. 1--9.
- Krizhevsky, A.; Sutskever, I. e Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*, pp. 1097--1105. ISSN 1090-0241.
- Laboratorio di Macchine Intelligenti per il riconoscimento di Video, Immagini e Audio (MIVIA) (2015). Fire Detection Dataset. <https://mivia.unisa.it/datasets/video-analysis-datasets/fire-detection-dataset/>. Acesso em 3 de Agosto de 2021.
- LeCun, Y. (1989). Generalization and Network Design Strategies. Relatório técnico, Department of Computer Science, University of Toronto, Toronto, Ontario, M5S 1A4, Canada.

- LeCun, Y.; Boser, B.; Denker, J.; Henderson, D.; Howard, R.; Hubbard, W. e Jackel, L. (1989a). Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, 2:396--404.
- LeCun, Y.; Jackel, L. D.; Boser, B.; Denker, J. S.; Graf, H. P.; Guyon, I.; Henderson, D.; Howard, R. E. e Hubbard, W. (1989b). Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning. *IEEE Communications Magazine*, 27(11):41--46.
- Li, H.; Kadav, A.; Durdanovic, I.; Samet, H. e Graf, H. P. (2016). Pruning Filters for Efficient ConvNets. *arXiv preprint arXiv:1608.08710*.
- Li, H.; Xu, Z.; Taylor, G.; Studer, C. e Goldstein, T. (2018). Visualizing the Loss Landscape of Neural Nets. Em *Advances in Neural Information Processing Systems*, pp. 6389--6399.
- Li, K.; Ma, W.; Sajid, U.; Wu, Y. e Wang, G. (2020). Object Detection with Convolutional Neural Networks. *Deep Learning in Computer Vision: Principles and Applications*, 30(31):41.
- Lin, M.; Chen, Q. e Yan, S. (2013). Network in Network. *arXiv preprint arXiv:1312.4400*.
- Lin, S.; Ji, R.; Li, Y.; Wu, Y.; Huang, F. e Zhang, B. (2018). Accelerating Convolutional Networks via Global & Dynamic Filter Pruning. Em *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2425--2432.
- Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B. e Belongie, S. (2017). Feature Pyramid Networks for Object Detection. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117--2125.
- Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P. e Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. Em *European conference on computer vision*, pp. 740--755. Springer.
- Liu, J.; Tripathi, S.; Kurup, U. e Shah, M. (2020). Pruning Algorithms to Accelerate Convolutional Neural Networks for Edge Applications: A Survey. *arXiv preprint arXiv:2005.04275*.
- Liu, S.; Qi, L.; Qin, H.; Shi, J. e Jia, J. (2018). Path Aggregation Network for Instance Segmentation. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8759--8768.

- Lloret, J.; Garcia, M.; Bri, D. e Sendra, S. (2009). A Wireless Sensor Network Deployment for Rural and Forest Fire Detection and Verification. *Sensors*, 9(11):8722--8747. ISSN 14248220.
- Loshchilov, I. e Hutter, F. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv preprint arXiv:1608.03983*.
- Lu, L.; Shin, Y.; Su, Y. e Karniadakis, G. E. (2019). Dying ReLU and Initialization: Theory and Numerical Examples. *arXiv*, 107:1--32.
- Lu, X.; Li, Q.; Li, B. e Yan, J. (2020). MimicDet: Bridging the Gap Between One-Stage and Two-Stage Object Detection. *arXiv preprint arXiv:2009.11528*.
- Lundrigan, P. (2018). Duplicate Image Finder. <https://git.io/JL0Qy>. Acesso em 2 de Fevereiro de 2020.
- Luo, J.-H. e Wu, J. (2017). An Entropy-based Pruning Method for CNN Compression. *arXiv preprint arXiv:1706.05791*.
- Luo, P.; Wang, X.; Shao, W. e Peng, Z. (2018). Towards Understanding Regularization in Batch Normalization. *arXiv preprint arXiv:1809.00846*.
- Luo, W.; Xing, J.; Milan, A.; Zhang, X.; Liu, W. e Kim, T.-K. (2020). Multiple Object Tracking: A Literature Review. *Artificial Intelligence*, p. 103448.
- Maas, A. L.; Hannun, A. Y. e Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 28.
- Marbach, G.; Loepfe, M. e Brupbacher, T. (2006). An Image Processing Technique for Fire Detection in Video Images. *Fire Safety Journal*, 41(4):285 – 289. ISSN 03797112.
- Mehmood, T.; Liland, K. H.; Snipen, L. e Sæbø, S. (2012). A Review of Variable Selection Methods in Partial Least Squares Regression. *Chemometrics and Intelligent Laboratory Systems*, 118:62--69.
- Mehrotra, K. G.; Mohan, C. K. e Huang, H. (2017). *Anomaly Detection Principles and Algorithms*. Springer.
- Metropolis, N. e Ulam, S. (1949). The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335--341.

- Mikołajczyk, A. e Grochowski, M. (2018). Data Augmentation for Improving Deep Learning in Image Classification Problem. Em *International Interdisciplinary PhD Workshop (IIPhDW)*, pp. 117--122. IEEE.
- Misra, D. (2019). Mish: A Self Regularized Non-Monotonic Neural Activation Function. *arXiv preprint arXiv:1908.08681*.
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York. ISBN 9780071154673.
- Mittal, D.; Bhardwaj, S.; Khapra, M. M. e Ravindran, B. (2018). Recovering from Random Pruning: On the Plasticity of Deep Convolutional Neural Networks. Em *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 848--857. IEEE.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D. e Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*.
- Mostofa, M.; Ferdous, S. N.; Riggan, B. S. e Nasrabadi, N. M. (2020). Joint-SRVDNet: Joint Super Resolution and Vehicle Detection Network. *IEEE Access*, 8:82306--82319.
- Muhammad, K.; Ahmad, J. e Baik, S. W. (2018). Early Fire Detection using Convolutional Neural Networks during Surveillance for Effective Disaster Management. *Neurocomputing*, 288:30--42. ISSN 18728286.
- Nair, V. e Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. Em *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10*, p. 807--814, Madison, WI, USA. Omnipress.
- National Fire Research Laboratory (NFRL) (2019). National Institute of Standards and Technology (NIST). <https://www.nist.gov/fire>. Acesso em 3 de Agosto de 2021.
- Obitko, M. e Jirkovský, V. (2015). Big Data Semantics in Industry 4.0. Em *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pp. 217--229. Springer.
- Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.;

- Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J. e Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Em Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. e Garnett, R., editores, *Advances in Neural Information Processing Systems 32*, pp. 8024--8035. Curran Associates, Inc.
- Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M. e Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825--2830.
- Petes, C. (2019). Africa is the 'fire continent' but blazes differ from Amazon. <https://bit.ly/3aSljQn>. Acesso em 24 de Dezembro de 2020.
- Pias, M.; Botelho, S. e Drews, P. (2019). Perfect Storm: DSAs Embrace Deep Learning for GPU-Based Computer Vision. Em *32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutoriais (SIBGRAPI-T)*, pp. 8--21. IEEE.
- Pilarczyk, R. e Skarbek, W. (2019). On Intra-Class Variance for Deep Learning of Classifiers. *arXiv preprint arXiv:1901.11186*.
- Polyak, B. T. (1964). Some Methods of Speeding up the Convergence of Iteration Methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1--17.
- Raghavendra, C. S.; Sivalingam, K. M. e Znati, T. (2006). *Wireless Sensor Networks*. Springer.
- Ramachandran, P.; Zoph, B. e Le, Q. V. (2018). Swish: A Self-Gated Activation Function. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, pp. 1--12.
- Raspberry Pi Foundation (2021). Raspberry Pi 4 Model B. <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>. Acesso em 9 de Maio de 2021.
- Rastegari, M.; Ordonez, V.; Redmon, J. e Farhadi, A. (2016). XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. Em *European Conference on Computer Vision*, pp. 525--542. Springer.
- Redmon, J. e Bochkovskiy, A. (2013). Darknet: Open Source Neural Networks in C. <https://github.com/alexeyab/darknet>. Acesso em 1 Janeiro de 2021.

- Redmon, J.; Divvala, S.; Girshick, R. e Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779--788.
- Redmon, J. e Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263--7271.
- Redmon, J. e Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.
- Reitermanová, Z. (2010). Data Splitting. Em *WDS'10 Proceedings of Contributed Papers*, volume 10, pp. 31--36.
- Ren, S.; He, K.; Girshick, R. e Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137--1149. ISSN 01628828.
- Ribani, R. e Marengoni, M. (2019). A Survey of Transfer Learning for Convolutional Neural Networks. Em *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pp. 47--57. IEEE.
- Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological review*, 65(6):386.
- Rumelhart, D. E.; Hinton, G. E. e Williams, R. J. (1986). Learning Representations by Back-Propagating Errors. *Nature*, 323(6088):533--536.
- Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. et al. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211--252.
- Russell, S. e Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, 3ª edição.
- Sadeghi, M. A. e Forsyth, D. (2014). 30Hz Object Detection with DPM V5. Em *European Conference on Computer Vision*, pp. 65--79. Springer.
- Saeed, F.; Paul, A.; Karthigaikumar, P. e Nayyar, A. (2019). Convolutional Neural Network based Early Fire Detection. *Multimedia Tools and Applications*, pp. 1--17.
- Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210--229.

- Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A. e Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510--4520.
- Santurkar, S.; Tsipras, D.; Ilyas, A. e Maşdry, A. (2018). How Does Batch Normalization Help Optimization? Em *Proceedings of the 32nd international conference on neural information processing systems*, pp. 2488--2498.
- Schultze, T.; Kempka, T. e Willms, I. (2006). Audio-Video Fire-Detection of Open Fires. *Fire Safety Journal*, 41(4):311--314. ISSN 03797112.
- Shi, L.; B, F. L.; Lin, C. e Zhao, Y. (2017). Video-Based Fire Detection with Saliency Detection and Convolutional Neural Networks. Em *Advances in Neural Networks, ISNN 2017*, volume 10878, pp. 299--309. Springer.
- Shorten, C. e Khoshgoftaar, T. M. (2019). A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60.
- Singh, A. K. e Singh, H. (2012). Forest Fire Detection through Wireless Sensor Network using Type-2 Fuzzy System. *International Journal of Computer Applications*, 52(9):975--8887.
- Srivastava, R. K.; Greff, K. e Schmidhuber, J. (2015). Highway Networks. *arXiv preprint arXiv:1505.00387*.
- State Key Lab of Fire Science (SKLFS) (2012). Video smoke detection. <http://staff.ustc.edu.cn/~yfn/vsd.html>. Acesso em 3 de Agosto de 2021.
- Stauffer, C. e Grimson, W. E. (1999). Adaptive Background Mixture Models for Real-Time Tracking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2(c):246--252. ISSN 10636919.
- Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V. e Rabinovich, A. (2015). Going Deeper with Convolutions. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1--9.
- Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J. e Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818--2826.

- Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C. e Liu, C. (2018). A Survey on Deep Transfer Learning. Em *International Conference on Artificial Neural Networks*, pp. 270--279. Springer.
- Tan, M.; Pang, R. e Le, Q. V. (2020). EfficientDet: Scalable and Efficient Object Detection. Em *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781--10790.
- Tao, C.; Zhang, J. e Wang, P. (2016). Smoke Detection Based on Deep Convolutional Neural Networks. Em *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration, ICIICII 2016*. ISSN 0027-8424.
- Texas Instruments (2021). BeagleBoard. <https://beagleboard.org/>. Acesso em 23 de Agosto de 2021.
- Thom, M. e Palm, G. (2013). Sparse Activity and Sparse Connectivity in Supervised Learning. *Journal of Machine Learning Research*, 14(Apr):1091--1143.
- Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267--288.
- Tikhonov, A. (1963). On Solving Ill-Posed Problem and Method of Regularization. *Doklady Akademii Nauk USSR*, 153:501--504.
- Töreyn, B. U. (2018). Smoke Detection in Compressed Video. Em *Applications of Digital Image Processing XLI*, volume 10752, p. 1075232. International Society for Optics and Photonics.
- Töreyn, B. U.; Cinbis, R. G.; Dedeoglu, Y. e Çetin, A. E. (2007). Fire Detection in Infrared Video Using Wavelet Analysis. *Optical Engineering*, 46(6):067204. ISSN 0091-3286.
- Töreyn, B. U.; Dedeoglu, Y. e Çetin, A. E. (2006). Contour Based Smoke Detection in Video using Wavelets. Em *14th European Signal Processing Conference*, pp. 1--5. IEEE. ISSN 22195491.
- Töreyn, B. U.; Dedeoglu, Y. e Çetin, A. E. (2005). Wavelet Based Real-Time Smoke Detection in Video. Em *13th European Signal Processing Conference*, pp. 1--4. IEEE.
- Toulouse, T.; Rossi, L.; Campana, A.; Celik, T. e Akhloufi, M. A. (2017). Computer Vision for Wildfire Research: An Evolving Image Dataset for Processing and Analysis. *Fire Safety Journal*, 92:188--194.

- Urban, G.; Geras, K. J.; Kahou, S. E.; Aslan, O.; Wang, S.; Caruana, R.; Mohamed, A.; Philipose, M. e Richardson, M. (2016). Do Deep Convolutional Nets Really Need to be Deep and Convolutional? *arXiv preprint arXiv:1603.05691*.
- Van Engelen, J. E. e Hoos, H. H. (2020). A Survey on Semi-Supervised Learning. *Machine Learning*, 109(2):373--440.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł. e Polosukhin, I. (2017). Attention is All You Need. Em *Advances in Neural Information Processing Systems*, pp. 5998--6008.
- Veit, A. e Belongie, S. (2018). Convolutional Networks with Adaptive Inference Graphs. Em *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3--18.
- Veit, A.; Wilber, M. J. e Belongie, S. (2016). Residual Networks Behave Like Ensembles of Relatively Shallow Networks. Em Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I. e Garnett, R., editores, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.
- Wang, C.-Y.; Liao, H.-Y. M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W. e Yeh, I.-H. (2020). CSPNet: A New Backbone That Can Enhance Learning Capability of CNN. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 390--391.
- Wang, C.-Y.; Yeh, I.-H. e Liao, H.-Y. M. (2021). You Only Learn One Representation: Unified Network for Multiple Tasks. *arXiv preprint arXiv:2105.04206*.
- Wang, K.; Lin, L.; Lu, J.; Li, C. e Shi, K. (2015). PISA: Pixelwise Image Saliency by Aggregating Complementary Appearance Contrast Measures With Edge-Preserving Coherence. *IEEE Transactions on Image Processing*, 24(10):3019--3033.
- Widrow, B. e Hoff, M. E. (1960). Adaptive Switching Circuits. Relatório técnico, Stanford University, Stanford Electronics Laboratories.
- Wold, H. (1973). Nonlinear Iterative Partial Least Squares (NIPALS) Modelling: Some Current Developments. Em *Multivariate Analysis-III*, pp. 383--407. Elsevier.
- Woo, S.; Park, J.; Lee, J.-Y. e Kweon, I. S. (2018). CBAM: Convolutional Block Attention Module. Em *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3--19.

- Yao, Z.; Cao, Y.; Zheng, S.; Huang, G. e Lin, S. (2020). Cross-Iteration Batch Normalization. *arXiv preprint arXiv:2002.05712*.
- Yin, Z.; Wan, B.; Yuan, F.; Xia, X. e Shi, J. (2017). A Deep Normalization and Convolutional Neural Network for Image Smoke Detection. *IEEE Access*, 5:18429--18438. ISSN 21693536.
- Yu, C.; Mei, Z. e Zhang, X. (2013). A Real-Time Video Fire Flame and Smoke Detection Algorithm. Em *Procedia Engineering*, volume 62, pp. 891--898. ISSN 18777058.
- Yu, L.; Wang, N. e Meng, X. (2005). Real-Time Forest Fire Detection with Wireless Sensor Networks. Em *International Conference on Wireless Communications, Networking and Mobile Computing.*, volume 2, pp. 1214--1217. IEEE. ISSN 2161-9646.
- Yuen, W. W. e Chow, W. K. (2005). A Monte Carlo Approach for the Layout Design of Thermal Fire Detection System. *Fire Technology*, 41:93--104.
- Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J. e Yoo, Y. (2019). CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023--6032.
- Zhang, Q.; Xu, J.; Xu, L. e Guo, H. (2016). Deep Convolutional Neural Networks for Forest Fire Detection. Em *Proceedings of the 2016 International Forum on Management, Education and Information Technology Application*. Atlantis Press.
- Zhang, Q. X.; Lin, G. H.; Zhang, Y. M.; Xu, G. e Wang, J. J. (2018). Wildland Forest Fire Smoke Detection Based on Faster R-CNN using Synthetic Smoke Images. Em *Procedia Engineering*, volume 211, pp. 441--446. Elsevier. ISSN 18777058.
- Zhang, Z.; He, T.; Zhang, H.; Zhang, Z.; Xie, J. e Li, M. (2019). Bag of Freebies for Training Object Detection Neural Networks. *arXiv preprint arXiv:1902.04103*.
- Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R. e Ren, D. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. Em *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 12993--13000.
- Zhu, L. S. e Liu, W. (2020). XOR-Net: An Efficient Computation Pipeline for Binary Neural Network Inference on Edge Devices. Em *The 26th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*.