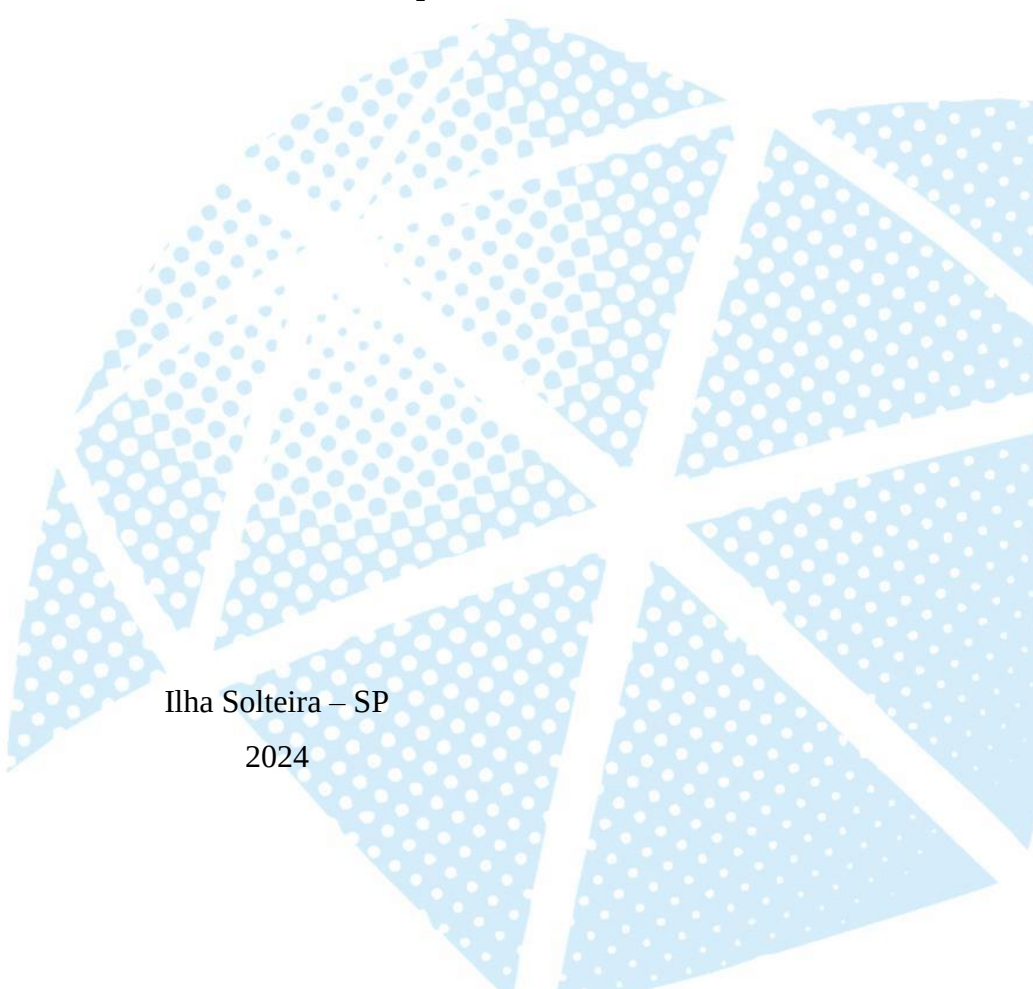


**UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
CÂMPUS DE ILHA SOLTEIRA**

RICARDO FERNANDO NUNES

**Classificação de Imagens Baseada no Aprendizado Profundo e na
Teoria da Ressonância Adaptativa**

Ilha Solteira – SP
2024



RICARDO FERNANDO NUNES

**CLASSIFICAÇÃO DE IMAGENS BASEADA NO APRENDIZADO PROFUNDO E
NA TEORIA DA RESSONÂNCIA ADAPTATIVA**

Tese apresentada à Faculdade de Engenharia
da Universidade Estadual Paulista – UNESP,
Câmpus de Ilha Solteira, como requisito para a
obtenção do título de Doutor em Engenharia
Elétrica. Área de Conhecimento: Automação.

Orientador: Prof. Dr. Carlos Roberto Minussi

Ilha Solteira - SP

2024

FICHA CATALOGRÁFICA

Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Nunes, Ricardo Fernando.
N972c Classificação de imagens baseada no aprendizado profundo e na teoria da
ressonância adaptativa / Ricardo Fernando Nunes. -- Ilha Solteira: [s.n.], 2024
114 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia
de Ilha Solteira. Área de conhecimento: Automação, 2024

Orientador: Carlos Roberto Minussi

Inclui bibliografia

1. Rede convolucional profunda. 2. Teoria da ressonância adaptativa. 3.
Classificação de imagem.


Amanda Sertori dos Santos

Bibliotecária - CRB/8-9061
Seção Técnica de Referência, Atendimento ao
Usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação

IMPACTO POTENCIAL DESTA PESQUISA

Esta pesquisa de doutorado excursionou adentro de uma fronteira do conhecimento humano nominada inteligência artificial (IA). Neste contexto, as grandes corporações usam, habitualmente, as inovações produzidas por pesquisadores e cientistas, em especial das universidades, aliada ao grande poder econômico, buscam monopolizar as aplicações nas várias áreas da atividade humana, tais como, os grandes modelos *chatbot*, nas comunicações, saúde, entre outras que representam vastos lucros e protagonismos. Invariavelmente, implantam unidades geradoras de eletricidade de grande porte para atender exclusivamente as suas demandas. Desenvolvem sistemas para atenderem os seus interesses em detrimento do que a sociedade efetivamente necessita. Usam mão de obra barata para treinarem as redes neurais artificiais. Contudo, diferentemente, há bastante espaço para o desenvolvimento de sistemas visando à resolução de problemas essencialmente de interesse social, tentando desvendar a complexidade da IA e transformando em algo tangível e útil de forma mais democrática possível. Este é o foco desta pesquisa. Por conseguinte, alternativamente aos modelos atuais de inteligência artificial, sobretudo no campo de visão computacional, amplamente utilizados na atualidade, essa pesquisa apresenta um estudo de um modelo alternativo que forneça maior confiabilidade e com desempenho competitivo em relação às técnicas atuais, que seja flexível de modo a viabilizar futuras implementações em diversas áreas, como: saúde, robótica, transporte, segurança.

POTENTIAL IMPACT OF THIS RESEARCH

This doctoral research explored a frontier of human knowledge called artificial intelligence (AI). In this context, large corporations habitually use innovations produced by researchers and scientists, especially from universities, combined with great economic power, they seek to monopolize applications in various areas of human activity, such as large chatbot models, in communications, health, among others that represent vast profits and prominence. They invariably deploy large electricity generating units to exclusively meet their demands. They develop systems to meet their interests to the detriment of what society actually needs. They use cheap labor to train artificial neural networks. However, differently, there is plenty of room for the development of systems aimed at solving problems essentially of social interest, trying to unravel the complexity of AI and transforming it into something tangible and useful in the most democratic way possible. This is the focus of this research. Therefore, as an alternative to current artificial intelligence models, especially in the field of computer vision, which are widely used today, this research presents a study of an alternative model that provides greater reliability and competitive performance in relation to current techniques, which is flexible in terms of in order to enable future implementations in various areas, such as: health, robotics, transport, security.

CERTIFICADO DE APROVAÇÃO

TÍTULO DA TESE: CLASSIFICAÇÃO DE IMAGENS BASEADA NO APRENDIZADO PROFUNDO E NA TEORIA DA RESSONÂNCIA ADAPTATIVA

AUTOR: RICARDO FERNANDO NUNES

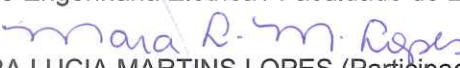
ORIENTADOR: CARLOS ROBERTO MINUSSI

Aprovado como parte das exigências para obtenção do Título de Doutor em Engenharia Elétrica, área: Automação pela Comissão Examinadora:


Prof. Dr. CARLOS ROBERTO MINUSSI (Participação Presencial)
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira - UNESP

Prof. Dra. ANNA DIVA PLASENCIA LOTUFO (Participação Presencial)
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira - UNESP

Documento assinado digitalmente
ANNA DIVA PLASENCIA LOTUFO
Data: 05/08/2024 06:58:27-0300
Verifique em <https://validar.iti.gov.br>


Prof. Dra. MARA LUCIA MARTINS LOPES (Participação Presencial)
Departamento de Matemática / Faculdade de Engenharia de Ilha Solteira - UNESP

Prof. Dr. KENJI NOSE FILHO (Participação Virtual)
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas / Universidade Federal de Minas Gerais

Documento assinado digitalmente
KENJI NOSE FILHO
Data: 02/08/2024 17:54:35-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. JOSÉ GUILHERME MAGALINI SANTOS DECANINI (Participação Virtual)
Departamento de Eletrotécnica / Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP

Documento assinado digitalmente
JOSE GUILHERME MAGALINI SANTOS DECANINI
Data: 02/08/2024 18:10:00-0300
Verifique em <https://validar.iti.gov.br>

Ilha Solteira, 02 de agosto de 2024

AGRADECIMENTOS

O início e finalização deste trabalho só foi possível pelo incentivo, apoio e colaboração de diferentes pessoas e instituições que cruzaram o meu caminho formativo ao longo da minha trajetória acadêmico-profissional.

Em especial, agradeço:

Aos meus pais, Marcos Luiz Nunes e Conceição de Souza Silva Nunes, por todo incentivo e apoio ao longo da minha trajetória de vida; sou grato a cada conselho, a cada suor e luta para conseguir dar educação aos filhos mesmo diante de tantas adversidades sociais e econômicas.

Ao meu irmão, Eduardo Fernando Nunes, meu parceiro desde sempre; em que dividimos tudo, desde balas e chocolates, mas, principalmente histórias, experiências e expectativas de vida.

À minha companheira de vida, Thamiris Silva Fernandes; que esteve ao meu lado durante a trajetória do doutorado, muitas vezes oferecendo ouvidos, ombros e abraços nos momentos de dúvidas e frustrações, mas também nos de alívio e felicidade.

A todos os professores que cruzaram minha trajetória formativa, da escola à universidade.

Aos docentes e funcionários do Departamento de Engenharia Elétrica e da Seção de Pós-Graduação da FEIS/UNESP, pelos ensinamentos ministrados e cordialidade.

Ao meu orientador, Prof. Dr. Carlos Roberto Minussi, por todas as oportunidades dadas a mim ao longo destes anos. Oportunidade de aprender, de crescer e de me reinventar a cada reunião. Oportunidade de partilhar comigo todo o conhecimento e sua dedicação em prol da ciência.

A todos os amigos que fiz durante o curso que diretamente ou indiretamente colaboraram para a realização deste trabalho, pelas trocas de conhecimento e boas recordações.

A todos os companheiros do Laboratório de Sistemas Inteligentes, em especial, ao Haislan Ranelli Santana Bernardes e ao Anthony Ferreira Lamarca, compartilhando conhecimentos, apoios, frustrações e boas notícias.

Ao Instituto Federal de Educação, Ciência e Tecnologia do Estado de São Paulo (IFSP) pelo incentivo acadêmico.

RESUMO

Com o recente desenvolvimento de grandes bancos de dados e do processamento paralelo, entre outras facilidades computacionais, as Redes Convolucionais Profundas, com sua alta precisão de classificação, ganharam muitos destaques em tarefas de visão computacional. Estas redes retêm uma grande quantidade de conhecimento, pois, normalmente, são treinadas de forma otimizada com muitas imagens, o que demanda um longo período. Entretanto, em suas formas clássicas, utilizam o algoritmo de retropropagação e, portanto, apresentam fragilidades como, esquecimento catastrófico, alucinação e passividade a ataques adversários. Por outro lado, as técnicas de transferência de conhecimento possibilitam aproveitar o vasto conhecimento retido nestas redes e utilizar, por exemplo, uma rede neural da família da Teoria da Ressonância Adaptativa como um agente classificador. Este estudo propõe um modelo, denominado de VGG ARTMAP *Fuzzy*, que utiliza a arquitetura VGG-16 para extração de características de imagens e uma rede neural ARTMAP *Fuzzy* para a classificação. O modelo é testado por meio de simulações com bancos de dados relativos à diferentes aplicações de problemas de visão computacional e seu desempenho é validado tanto pela acurácia de suas respostas em testes quanto aos tempos de treinamento e teste.

Palavras-chave: rede convolucional profunda; teoria da ressonância adaptativa; classificação de imagem.

ABSTRACT

With the recent development of large databases and parallel processing, among other computational facilities, Deep Convolutional Networks, with their high classification accuracy, have gained much prominence in computer vision tasks. These networks retain a large amount of knowledge because they are usually trained in an optimized way with many images, which requires a long period of time. However, in their classical forms, they use the backpropagation algorithm and, therefore, present weaknesses such as catastrophic forgetting, hallucination and passivity to adversarial attacks. On the other hand, knowledge transfer techniques make it possible to take advantage of the vast knowledge retained in these networks and use, for example, a neural network from the Adaptive Resonance Theory family as a classifying agent. This study proposes a Fuzzy VGG ARTMAP model, which uses the VGG-16 architecture for image feature extraction and a Fuzzy ARTMAP neural network for classification. The model is tested through simulations with databases related to different applications of computer vision problems and its performance is validated both by the accuracy of its responses in tests and by the training and testing times.

Keywords: deep learning network, adaptive resonance theory, image classification.

LISTA DE FIGURAS

Figura 1 - Inteligência artificial, aprendizado de máquina e aprendizado profundo.....	15
Figura 2 - Demonstração de um ataque de exemplo adversário.....	19
Figura 3 - Arquitetura básica de uma rede neural convolucional.....	25
Figura 4 - Operação de uma CNN para classificação de imagens.....	26
Figura 5 - Processamento de uma imagem pelo processo de convolução.....	27
Figura 6 - Ilustração de duas camadas convolucionais.....	28
Figura 7 - Exemplos de campos receptíveis.	29
Figura 8 - Diferentes tipos de tratamento da borda da imagem durante a convolução: (a) valor zero assumido; (b) adjacência considerada; (c) borda desconsiderada.	31
Figura 9 - Convolução com <i>stride</i> com valor 1 e sem <i>padding</i>	31
Figura 10 - Exemplos de funções de ativações usadas em redes neurais profundas.....	33
Figura 11 - Operações de <i>pooling</i> : tamanho 2x2 e <i>stride</i> 2.....	34
Figura 12 - Arquitetura LeNet-5.....	36
Figura 13 - Arquitetura AlexNet.	37
Figura 14 - Comparação entre camadas convolucionais: (a) tradicional em CNN e (b) <i>mlpconv</i>	38
Figura 15 - Comparação entre camadas totalmente conectadas: (a) <i>flatten</i> e (b) <i>global average pooling</i>	39
Figura 16 - Arquitetura da <i>Network in Network</i>	39
Figura 17 - Arquitetura VGGnet-16 (configuração-D).	41
Figura 18 - Estrutura do Módulo <i>Inception</i> : (a) Conceito básico do módulo <i>Inception</i> (b) Módulo <i>Inception</i> com redução de dimensionalidade.....	42
Figura 19 - Arquitetura rede GoogLeNet.	43
Figura 20 - Bloco residual de duas camadas.	44
Figura 21 - Arquiteturas de blocos residuais: (a) comum; (b) <i>Bottleneck</i>	46
Figura 22 - Comparação entre exemplos de blocos residuais de dimensão 256: (a) ResNet; (b) ResNext.	47
Figura 23 - Conexões entre blocos densos: etapa de propagação.....	48
Figura 24 - Comparação entre aprendizados: (a) tradicional; (b) por transferência.....	53
Figura 25 - Representação do aprendizado de transferência a partir da definição de domínio e tarefa.	54
Figura 26 - Abordagens de transferência de aprendizado para CNNs.	58
Figura 27 - Anatomia da rede ART elementar.	64

Figura 28 - Fluxograma do algoritmo de treinamento da rede ART 1.....	68
Figura 29 - Classificação por hiper-retângulo de uma categoria.....	69
Figura 30 - Anatomia e algoritmo de treinamento da rede ART <i>Fuzzy</i>	70
Figura 31 - Arquitetura básica das redes ARTMAP.....	73
Figura 32 - Algoritmo da rede ARTMAP <i>Fuzzy</i>	75
Figura 33 - Arquitetura VGG-ARTMAP <i>Fuzzy</i> : versão com achatamento.	78
Figura 34 - Arquitetura VGG-ARTMAP <i>Fuzzy</i> : versão com <i>pooling</i> médio global.	78
Figura 35 - VGG 16 adaptada com camada densa.	80
Figura 36 - MNIST: Exemplos de amostras ambíguas.....	81
Figura 37 - VGG ARTMAP <i>Fuzzy</i> com achatamento - relação do número de categorias da ARTa por amostra para o conjunto MNIST.	83
Figura 38 - VGG ARTMAP <i>Fuzzy</i> com achatamento - relação do tempo de treinamento por amostra para o conjunto MNIST.	83
Figura 39 - VGG ARTMAP <i>Fuzzy</i> com achatamento - Desempenho do sistema para a etapa de teste com o conjunto MNIST.....	84
Figura 40 - VGG ARTMAP <i>Fuzzy</i> com <i>pooling</i> médio global - relação do número de categorias da ARTa por amostra para o conjunto MNIST.	86
Figura 41 - VGG ARTMAP <i>Fuzzy</i> com <i>pooling</i> médio global - relação do tempo de treinamento por amostra para o conjunto MNIST.....	86
Figura 42 - VGG ARTMAP <i>Fuzzy</i> com <i>pooling</i> médio global - Desempenho do sistema para a etapa de teste com o conjunto MNIST.	87
Figura 43 - VGG-16 curvas de acurácias de treinamento e validação por época.....	88
Figura 44 - VGG 16 adaptada - Desempenho do sistema para a etapa de teste com o conjunto MNIST.....	89
Figura 45 - Exemplos de amostras adversárias do MNIST.	91
Figura 46 - VGG-16 adaptada - Desempenho do sistema para a etapa de teste com o conjunto MNIST adversário.	92
Figura 47 - VGG ARTMAP <i>Fuzzy</i> com achatamento - Desempenho do sistema para a etapa de teste com o conjunto MNIST adversário.	93
Figura 48 - Exemplos de amostras: <i>Brain MRI Images for Brain Tumor Detection</i>	94
Figura 49 - Exemplos de amostras: <i>Car vs Bike Classification</i>	95
Figura 50 - Exemplos de amostras: <i>Muffin vs chihuahua</i>	96
Figura 51 - Exemplos de amostras: <i>Ripe and Unripe Tomatoes</i>	96
Figura 52 - Exemplos de amostras: <i>Rock-Paper-Scissors game</i>	97
Figura 53 - Exemplos de amostras: <i>Traffic Sign</i>	98

LISTA DE TABELAS

Tabela 1 - Arquiteturas de ResNet para ImageNet.....	45
Tabela 2 - Arquiteturas de DenseNets.....	50
Tabela 3 - Relação entre as diferentes configurações de aprendizagem de transferência.....	56
Tabela 4 - Diferentes abordagens usadas em diferentes configurações.....	57
Tabela 5 - Exemplo de processamento por <i>One-Hot-Encoding</i>	79
Tabela 6 - Dados de treinamento e teste do conjunto MNIST - VGG ARTMAP <i>Fuzzy</i> com achatamento.....	82
Tabela 7 - Dados de treinamento e teste do conjunto MNIST - VGG ARTMAP <i>Fuzzy</i> com <i>pooling</i> médio global.....	85
Tabela 8 - Dados de treinamento e teste do conjunto MNIST - VGG adaptada.....	88
Tabela 9 - Comparação de acurácias para MNIST.....	90
Tabela 10 - Acurácias e tempo de treinamento de VGG ARTMAP <i>Fuzzy</i> com <i>pooling</i> médio global e VGG-16 adaptada.....	99

LISTA DE ABREVIATURAS E SIGLAS

ANN	<i>Artificial Neural Network</i>
ART	<i>Adaptive Resonance Theory</i>
AE	<i>Auto-Encoder</i>
BIM	<i>Basic Iterative Method</i>
CNN	<i>Convolutional Neural Network</i>
DBN	<i>Deep Belief Network</i>
DL	<i>Deep Learning</i>
DNN	<i>Deep Neural Network</i>
DRL	<i>Deep Reinforcement Learning</i>
ERF	<i>Effective Receptive Field</i>
FC	<i>Fully Connected</i>
FCN	<i>Fully Convolutional Network</i>
FGSM	<i>Fast Gradient Sign Method</i>
GAN	<i>Generative Adversarial Network</i>
GPU	<i>Graphics Processing Unit</i>
GRU	<i>Gated Recurrent Units</i>
IA	<i>Inteligência Artificial</i>
ILSVRC	<i>ImageNet LargeScale Visual Recognition Challenge</i>
LSTM	<i>Long Short-Term Memory</i>
LTM	<i>Long-Term Memory</i>
ML	<i>Machine Learning</i>
MCP	<i>McCulloch-Pitts</i>
NiN	<i>Network in Network</i>
PGD	<i>Projected Gradient Descent</i>
ReLU	<i>Rectified Linear Activation Function</i>
RF	<i>Receptive Field</i>
RNA	<i>Redes Neurais Artificiais</i>
RNN	<i>Recurrent Neural Networks</i>
STM	<i>Short-Term Memory</i>
VG	<i>Vanishing Gradient</i>
VGG	<i>Visual Geometry Group</i>
WTA	<i>Winner Take All</i>

SUMÁRIO

1	INTRODUÇÃO	15
1.1	A PROBLEMÁTICA DO APRENDIZADO PROFUNDO	18
1.2	ALTERNATIVAS AO APRENDIZADO PROFUNDO	20
1.3	OBJETIVOS	21
1.4	JUSTIFICATIVA	22
1.5	ORGANIZAÇÃO DO TEXTO	22
2	REDES NEURASIS CONVOLUCIONAIS	24
2.1	ARQUITETURA DA REDE NEURAL CONVOLUCIONAL	24
2.1.1	Camada convolucional	26
2.1.1.1	Mapas de caraterísticas	27
2.1.1.2	Campo receptível	29
2.1.1.3	Padding	30
2.1.1.4	Stride.....	31
2.1.1.5	Função de ativação	32
2.1.2	Camada de pooling	33
2.1.3	Camada totalmente conectada	34
2.2	ARQUITETURAS POPULARES DE CNNs.....	35
2.2.1	LeNet.....	35
2.2.2	AlexNet.....	36
2.2.3	Network in Network	37
2.2.4	VGGnet.....	40
2.2.5	GoogLeNet.....	41
2.2.6	ResNet	44
2.2.7	ResNext	46
2.2.8	DenseNet	48
3	TRANSFERÊNCIA DE APRENDIZADO	51

3.1	APRENDIZADO TRADICIONAL E APRENDIZADO TRANSFERIDO	52
3.2	DEFINIÇÕES E NOTAÇÕES	53
3.3	CONFIGURAÇÕES DE TRANSFERÊNCIA DE APRENDIZADO	54
3.3.1	Transferência de conhecimento indutiva	54
3.3.2	Transferência de conhecimento transdutiva	55
3.3.3	Transferência de conhecimento não supervisionada	56
3.4	ABORDAGENS DE APRENDIZAGEM DE TRANSFERÊNCIA	56
3.5	TRANSFERÊNCIA DE APRENDIZADO PARA REDES CONVOLUCIONAIS.	58
3.5.1	Estratégias de transferência de aprendizado	59
3.6	TRANSFERÊNCIA NEGATIVA	60
4	TEORIA DA RESSONÂNCIA ADAPTATIVA	62
4.1	CARACTERÍSTICAS E DINÂMICA DAS REDES ART	63
4.2	ART 1	65
4.3	ART <i>FUZZY</i>	68
4.4	ARTMAP E ARTMAP <i>FUZZY</i>	72
5	VGG ARTMAP <i>FUZZY</i>	76
6	SIMULAÇÕES E RESULTADOS	80
6.1	MNIST	81
6.1.1	VGG ARTMAP Fuzzy com achatamento	81
6.1.2	VGG ARTMAP Fuzzy com pooling médio global	84
6.1.3	VGG-16 adaptada	87
6.1.4	Considerações sobre os resultados: MNIST	89
6.2	MNIST ADVERSÁRIO	90
6.2.1	VGG-16 adaptada	91
6.2.2	VGG ARTMAP Fuzzy com achatamento	92
6.2.3	Considerações sobre os resultados: MNIST adversário	93

6.3	OUTROS BANCOS DE DADOS	94
6.3.1	Brain MRI Images for Brain Tumor Detection	94
6.3.2	Car vs Bike Classification	95
6.3.3	Muffin vs chihuahua.....	95
6.3.4	Ripe and Unripe Tomatoes	96
6.3.5	Rock-Paper-Scissors game.....	96
6.3.6	Traffic Sign.....	97
6.3.7	Considerações sobre os resultados: outros bancos de dados	98
7	CONSIDERAÇÕES FINAIS.....	100
7.1	CONSIDERAÇÕES GERAIS	100
7.2	VARIANTES.....	101
7.3	PROCESSAMENTO COMPUTACIONAL	102
7.4	ATAQUES ADVERSÁRIOS.....	102
7.5	SUGESTÕES PARA TRABALHOS FUTUROS.....	103
	REFERÊNCIAS.....	106

1 INTRODUÇÃO

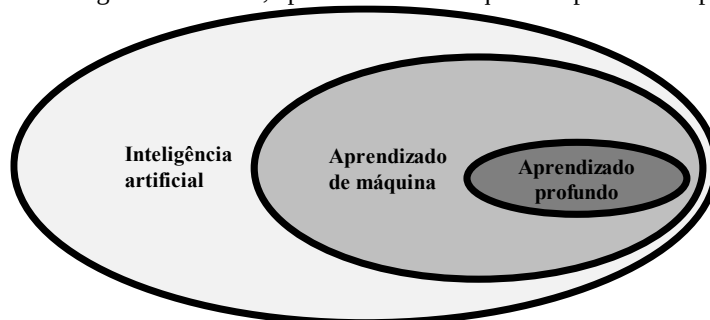
O aprendizado de máquina, popularmente conhecido como *Machine Learning* (ML), é um campo de pesquisa que utiliza algoritmos para extrair informações de dados brutos e representá-los por meio de algum tipo de modelo matemático. Várias ciências estão associadas ao aprendizado de máquina, incluindo inteligência artificial (IA), psicologia e teoria da informação Alom *et al.* (2019); Zhang *et al.* (2021).

O aprendizado de máquina possui a tarefa de identificar padrões de exemplos de treinamento e aplicar esses padrões (ou representações) aprendidos a novos dados não vistos. Por isto, às vezes é chamado de aprendizado superficial, por causa da sua natureza de aprender representações de camada única ou poucas camadas - na maioria dos casos (Dong; Wang; Abbas, 2021).

A eficiência dos algoritmos de aprendizado de máquina depende muito da qualidade da representação dos dados de entrada. Uma representação de dados ruim geralmente leva a degradação do desempenho. Deste modo, a construção de recursos a partir de dados brutos, conhecida como engenharia de recursos, é fundamental e, portanto, uma importante direção de pesquisa em aprendizado de máquina há muito tempo (Goodfellow; Bengio; Courville, 2016; Pan; Yang, 2010).

A engenharia de recursos (ou atributos), normalmente, é muito específica em seu domínio e requer um esforço humano intenso. Comparativamente, algoritmos de aprendizado profundo - *Deep Learning* (DL) - realizam extração de recursos de maneira automática, o que permite aos pesquisadores extrair recursos discriminativos com pouco conhecimento de domínio e esforço humano mínimo. Como ilustrado na Figura 1, o aprendizado profundo é um subcampo do aprendizado de máquina que, por sua vez, é um subcampo da inteligência artificial (Alom *et al.*, 2019).

Figura 1 - Inteligência artificial, aprendizado de máquina e aprendizado profundo.



Fonte: Elaboração do próprio autor.

A história do aprendizado de máquina e, conseqüentemente, do aprendizado profundo, começou em 1943, quando o modelo McCulloch-Pitts (MCP) foi introduzido e ficou

conhecido como o protótipo do modelo de um neurônio artificial (McCulloch; Pitts, 1943). Após o modelo MCP, a teoria Hebbiana, originalmente utilizada para os sistemas biológicos no ambiente natural, foi implementada (Hebb, 1949). Em seguida, o primeiro modelo matemático de um neurônio biológico chamado “*perceptron*”, no contexto do sistema de cognição, foi introduzido em 1958 (Rosenblatt, 1958). No final do primeiro inverno da inteligência artificial, Werbos introduziu o algoritmo retropropagação (*backpropagation*) (Werbos, 1974) e o uso de erros no treinamento de modelos de aprendizado, abriu as portas para as redes neurais modernas. Em 1980, o “*neocogitron*”, que inspirou a rede neural convolucional, foi introduzido (Fukushima; Miyake, 1982), enquanto as Redes Neurais Recorrentes (RNNs) foram propostas em 1986 (Jordan, 1986). Em seguida, LeNet fez uma Rede Neural Profunda (DNN) básica funcionar na década de 1990 (LeCun *et al.*, 1998). Esta rede foi fundamental para o surgimento de redes profundas muito utilizadas no campo de visão computacional - as Redes Neurais Convolucionais (CNN).

Apesar de técnicas de inteligência artificial existirem, pelo menos, desde a década de 1950, as arquiteturas desses modelos estagnaram por muito tempo por causa da falta de recursos que os fizessem realmente disseminar, como o processamento de *hardware*, por exemplo. Foi a partir do desenvolvimento do *Big Data* e do processamento paralelo em GPU's que foi possível processar grandes quantidades de dados em grandes modelos de aprendizado de máquinas. A disponibilidade e o crescimento explosivo de dados, além do notável avanço nas tecnologias de *hardware*, levaram ao surgimento de novos estudos em aprendizado distribuído e profundo.

Originado das Redes Neurais Artificiais (RNAs) e após décadas de desenvolvimento, o aprendizado profundo evoluiu como uma estrutura forte e eficiente que pode ser aplicada a um amplo espectro de problemas complexos de aprendizado, que eram difíceis de resolver usando as técnicas tradicionais de aprendizado de máquina no passado. O avanço do aprendizado profundo foi tão radical que hoje pode superar o desempenho em nível humano em classificação de imagens, por exemplo Deng *et al.* (2009); Russakovsky *et al.* (2015a).

O aprendizado profundo permite que modelos computacionais compostos por várias camadas de processamento e com alto nível de abstração aprendam com a experiência e percebam o mundo em termos de hierarquia de conceitos. Deste modo, as redes profundas consistem em várias camadas empilhadas uma após a outra e capturam representações sucessivas.

A guinada quando se trata de aprendizado profundo é como se aprende uma função $f(\cdot)$. De modo geral, modelos não profundos - comumente referidos na literatura como

“superficiais” (do termo em inglês: *shallow*) - buscam diretamente por uma função que possa, a partir de um conjunto de parâmetros, gerar o resultado desejado. Por outro lado, o aprendizado profundo utiliza métodos que aprendem $f(\cdot)$ por meio de composições de funções, como exemplificadas na Equação (1).:

$$f(x) = f_l((\dots f_2(f_1(x_1, W_1), W_2) \dots), W_l) \quad (1)$$

De modo que, o índice l se refere a uma camada; a função $f_l(\cdot)$ toma como entrada um vetor de dados x_l e gera, como saída, o próximo vetor: x_{l+1} ; W_l é o conjunto de parâmetros que cada função utiliza para realizar a transformação dos dados de entrada.

Diferentes métodos de aprendizado compõe o aprendizado profundo, incluindo aprendizado supervisionado, semissupervisionado e não supervisionado. Estes métodos são aplicados a um amplo espectro de tarefas complexas de aprendizado de máquina, *e.g.*: visão computacional (Krizhevsky; Sutskever; Hinton, 2012; LeCun *et al.*, 1998; Simonyan; Zisserman, 2014), reconhecimento de fala (Hinton, Geoffrey *et al.*, 2012), processamento de linguagem natural (Sutskever; Vinyals; Le, 2014), saúde (Chen *et al.*, 2017), robótica (Siau; Wang, 2018) e muitos outros.

As técnicas de aprendizado profundo também são vastas, dentre as mais comuns, cita-se: *Deep Neural Network* (DNN), Rede Neural Convolucional (CNN), Rede Neural Recorrente (RNN), *Long Short-Term Memory* (LSTM), *Gated Recurrent Units* (GRU), *Auto-Encoder* (AE), *Deep Belief Network* (DBN), *Generative Adversarial Network* (GAN) e *Deep Reinforcement Learning* (DRL).

O aprendizado profundo, com suas altas precisões em diversas tarefas, trouxe uma grande revolução nos serviços baseados em inteligência artificial na internet, incluindo serviços de IA baseados na computação de empresas em nuvem. O uso extensivo de aplicativos baseados em aprendizado profundo pode ser visto em ambientes críticos de segurança, como detecção de *malware*, carros autônomos, drones e robótica (Alom *et al.*, 2019; Dong; Wang; Abbas, 2021). Com os recentes avanços nos sistemas de reconhecimento facial, caixas eletrônicos e telefones celulares usam autenticação biométrica como recurso de segurança; sistemas controláveis por voz e modelos de reconhecimento automático de fala tornaram possíveis produtos como Amazon Alexa, Apple Siri e Microsoft Cortana (Chakraborty *et al.*, 2021).

1.1 A PROBLEMÁTICA DO APRENDIZADO PROFUNDO

As redes neurais profundas, em suas formas clássicas, são definidas por uma arquitetura *feedforward* que utiliza o algoritmo de retropropagação com aprendizado supervisionado. Deste modo, os pesos são alterados quando, em resposta a um vetor de entrada, seu filtro adaptativo gera um vetor de saída que não corresponde ao vetor de saída desejado. Consequentemente, o aprendizado profundo compartilha as mesmas características computacionais do algoritmo de retropropagação: o ajuste de peso não local durante o aprendizado por erro (Grossberg, 2020).

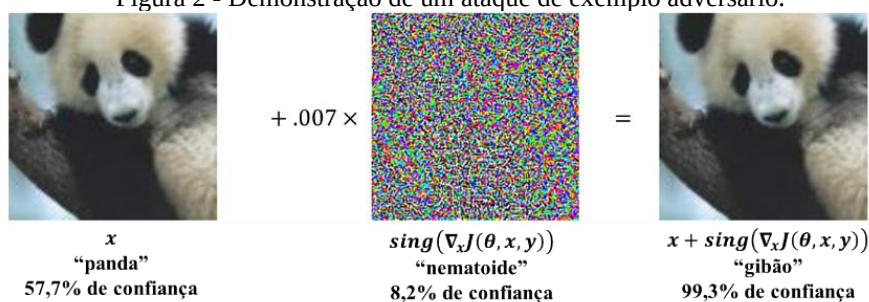
Um dos problemas do aprendizado profundo está relacionado ao seu método de treinamento, que pode experimentar o esquecimento catastrófico, *i.e.*, em qualquer estágio do aprendizado, uma parte imprevisível de sua memória pode entrar em colapso (McCloskey; Cohen, 1989). Além disto, a computação resultante - descoberta automaticamente por retropropagação via aprendizado supervisionado - pode ser difícil de interpretar e ter propriedades contraintuitivas (Goodfellow; Shlens; Szegedy, 2014).

Na conjuntura de algoritmos especializados em visão computacional, Szegedy, *et al.* (2013) revelaram uma outra fraqueza intrigante das redes neurais profundas: as redes profundas modernas, apesar de suas altas precisões, são surpreendentemente suscetíveis a ataques adversários (do inglês *Adversarial Attacks*). Ataques adversários – muitas vezes abordados como exemplos adversários - são amostras projetadas com adição de pequenas perturbações para provocar erros em classificações.

Na Figura 2 é representado um exemplo de ataque por uma imagem adversária, apresentada em Goodfellow; Shlens; Szegedy (2014). Ao olho humano, muitas vezes, estas perturbações são quase imperceptíveis, ou até mesmo invisíveis. Entretanto, para uma rede neural, são capazes de produzir uma classificação equivocada com a confiança significativamente alta, *i.e.*, as redes profundas preveem a classe errada com maior confiança do que todas as outras classes.

Existem vários métodos de como adicionar ruídos às imagens para criarem ataques adversários, destacam-se Su; Vargas; Sakurai, (2019) que foram capazes de criar um ruído de apenas um pixel que ocasiona erros na classificação quando se usa as redes profundas. Para este propósito, alteraram pontos específicos de dados localizados próximos aos limites de decisão que alteram as classificações das classes.

Figura 2 - Demonstração de um ataque de exemplo adversário.



Fonte: Adaptado de: Goodfellow; Shlens; Szegedy (2014).

Um ataque adversário pode conter vários objetivos, *e.g.*: a redução de confiança do classificador, o classificador produz a previsão correta, porém com menor confiança; um ataque não direcionado, o classificador gera qualquer classe além da previsão correta; ou ainda um ataque direcionado, quando o objetivo é que o classificador resulte uma classificação específica diferente da correta (Costa *et al.*, 2024).

Outra característica importante dos ataques adversários é que podem ser classificados de acordo com a quantidade de conhecimento ao qual o invasor tem acesso, *i.e.*, são considerados ataques caixa branca quando o atacante tem acesso à arquitetura ou aos dados; ou ataques caixa preta, quando o atacante consegue apenas acessar a previsão ou pares de entrada e saída da arquitetura.

Muitas pesquisas relacionadas a ataques adversários, bem como suas contramedidas, surgiram desde que Szegedy, *et al.* (2013) demonstraram a vulnerabilidade dos algoritmos de aprendizado de máquina (Akhtar; Mian, 2018; Bai *et al.*, 2021; Chakraborty *et al.*, 2021). Todavia, foi com a migração das redes profundas dos laboratórios para o mundo real que despontou preocupações sobre a segurança e a integridade de aplicações dependentes destes algoritmos.

Kurakin; Goodfellow; Bengio (2018) foram os primeiros a demonstrar que ataques adversários no mundo físico, *i.e.*, exemplos adversários impressos que podem ser classificados incorretamente, quando visualizados por meio da câmera de um *smartphone*. Posteriormente, (Eykholt *et al.*, 2018) alcançam altas taxas de classificações incorretas ao criar diferentes perturbações em placas de sinais de trânsito reais.

Os ataques adversários, em muitos casos, são concentrados em algoritmos baseados em gradiente descendente e utilizam perturbações para maximizar o erro de redes profundas, *e.g.*, Método de Sinal de Gradiente Rápido, ou *Fast Gradient Sign Method* (FGSM) (Goodfellow; Shlens; Szegedy, 2014), Método Iterativo Básico, ou *Basic Iterative Method*

(BIM) (Kurakin; Goodfellow; Bengio, 2016a) e ataque de Descida de Gradiente Projetada, *Projected Gradient Descent* (PGD) (Madry *et al.*, 2017).

Por serem treinadas por retropropagação, além de intrinsicamente possuírem pontos cegos e características não intuitivas, estes estudos expõem uma grande vulnerabilidade destes algoritmos. Deste modo, para Grossberg, (2020), apesar de técnicas de aprendizado profundo fazerem classificações com alta precisão, elas não são totalmente explicáveis e, portanto, não podem ser usadas com plena confiança.

De forma complementar, no que diz respeito ao seu treinamento, o aprendizado profundo tem outra grande falha prática: exige tempo e trabalho não apenas para coletar uma grande quantidade de dados, mas também para rotulá-los. Neste aspecto, as áreas de aprendizagem semissupervisionada, aprendizagem contínua, aprendizagem por transferência e meta-aprendizagem têm se destacado.

1.2 ALTERNATIVAS AO APRENDIZADO PROFUNDO

Como consequência de seus bons resultados, atualmente, o aprendizado profundo é amplamente utilizado em diversas aplicações. Entretanto, diferentes tipos de ataques aproveitam as vulnerabilidades de redes profundas para comprometer um sistema de aprendizado. Portanto, é extremamente importante fornecer robustez aos algoritmos de aprendizado profundo.

Chakraborty, *et al.* (2021) citam que existem poucas contramedidas fortes que podem ser usadas para mitigar os efeitos de ataques adversários e projetar um sistema robusto de aprendizado profundo. Dentre as contramedidas propostas nos últimos anos: o uso da técnica de *Data Augmentation* no treinamento (Kurakin; Goodfellow; Bengio, 2016b), destilação (Hinton; Vinyals; Dean, 2015; Papernot; McDaniel; Wu; *et al.*, 2016) e rede adversária generativa (Samangouei; Kabkab; Chellappa, 2018). Entretanto, Chakraborty, *et al.* (2021) destacam que, embora as eficiências dos mecanismos propostos, eles são específicos e não poderiam ser usados como uma solução única para todos os tipos de ataques.

Para Bai, *et al.* (2021), entre as várias estratégias de defesa existentes, o treinamento adversário é uma das abordagens mais eficazes para modelos de aprendizado profundo se defender contra os ataques adversários. Ao contrário de outras estratégias de defesa, o treinamento adversário visa aumentar a robustez dos modelos intrinsecamente, deste modo, os modelos treinados adversamente têm melhores desempenhos quando enfrentam exemplos adversários aos comparados com modelos treinados de modo padrão. No entanto, o treinamento com exemplos adversários não é prático em razão da necessidade de otimização

restrita e computacionalmente custosa. Além disto, a implementação dessas estratégias de defesa pode levar à degradação do desempenho e eficiência do modelo em questão (Goodfellow; Shlens; Szegedy, 2014; Szegedy *et al.*, 2013).

Como supracitado, a maioria das técnicas de ataques adversários são fundamentadas em aprendizados baseados em gradiente descendente, a base do algoritmo de treinamento por retropropagação. Além disto, redes *feedforward* profundas possuem pontos cegos intrínsecos, não são explicáveis e, durante o treinamento, que necessita de aprendizado por experiência de erro, o ajuste de peso não local pode ocasionar esquecimento catastrófico. Sem garantias contra o esquecimento catastrófico, um algoritmo de aprendizado não é plenamente confiável (Grossberg, 2020).

Todas estas características e vulnerabilidades fortalecem a necessidade de investigação de novas alternativas, com algoritmos baseados em técnicas de redes neurais que não são de arquitetura *feedforward* profundas, tampouco são dependentes do algoritmo de retropropagação.

As redes baseadas na Teoria da Ressonância Adaptativa - *Adaptive Resonance Theory* (ART) superam os problemas computacionais de retropropagação e aprendizado profundo, pois são capazes de aprender - incrementalmente - padrões de entrada arbitrários de maneira rápida, auto-organizada e estável, refinar o conhecimento e não perder informações já aprendidas. As memórias aprendidas são dinamicamente autoestabilizadas para evitar o esquecimento catastrófico de informações previamente aprendidas, mas ainda são úteis em resposta ao aprendizado futuro. Estas características aludem ao dilema da plasticidade e estabilidade.

A teoria ART é flexível e, portanto, as arquiteturas de redes baseadas nesta teoria de aprendizado, a partir de técnicas de transferência de conhecimento, podem ser compatíveis com algoritmos baseados em aprendizado profundo (como as redes neurais convolucionais) e aproveitem as características visuais aprendidas dos densos conjuntos de dados de visão computacional. Desta forma, em conjunto com redes profundas já treinadas, as redes baseadas em ART podem ser utilizadas como classificadores.

1.3 OBJETIVOS

O objetivo deste estudo é propor um modelo híbrido, denominado de VGG ARTMAP Fuzzy, que aproveita a experiência de redes convolucionais, pela arquitetura de aprendizado profundo VGG-16 com pesos pré-treinados no banco de dados ImageNet e associa, pela

técnica de transferência de aprendizado, com uma rede neural ARTMAP *Fuzzy*, responsáveis pelas etapas de extração de característica e de classificação, respectivamente.

O modelo proposto é validado a partir de simulações de bancos de dados composto por diferentes tipos de imagens e seu comportamento é investigado pelos âmbitos tanto da acurácia quanto aos tempos de treinamentos e testes, uma vez que estes parâmetros também são significativos no campo de visão computacional.

1.4 JUSTIFICATIVA

Redes convolucionais que alcançam alto desempenho de ponta para diferentes tarefas de visão computacional retêm uma grande quantidade de conhecimento, pois, normalmente, são treinadas a partir de milhões de imagens, o que requer um alto poder computacional e um longo período para esta tarefa. Muito destes modelos pré-treinados estão disponíveis publicamente para *download*, permitindo o uso em diversas aplicações.

Com o uso de técnicas de transferência de aprendizado, redes baseadas na Teoria da Ressonância Adaptativa (ART) podem usufruir da vastidão de conhecimento adquirido por longos treinamentos de redes profundas e atuar como classificadores das características de imagens extraídas pelas CNNs.

Por respeitarem o dilema da estabilidade e plasticidade, as Redes ARTs possuem vantagens em relação às redes com camadas totalmente conectadas e podem ser utilizadas como classificadoras das características extraídas das camadas convolucionais. Além disto, ao contrário do algoritmo de retropropagação, que podem apresentar esquecimento catastrófico, estas redes permitem futuros treinamentos sem comprometer os conhecimentos anteriormente adquiridos.

1.5 ORGANIZAÇÃO DO TEXTO

O conteúdo desta pesquisa está dividido em sete capítulos. O Capítulo 1 introduziu sobre Machine Learning e Aprendizado Profundo e, além de seus atributos e qualidades, expos também suas vulnerabilidades que motivaram e justificaram a tese. Continuamente, os três capítulos subsequentes apresentam a tríade de fundamentos necessários para embasar a proposta.

No Capítulo 2 é apresentada a base das Redes Neurais Convolucionais, suas principais características e as propriedades das principais arquiteturas destas redes.

As definições e notações da transferência de aprendizado são apresentadas no Capítulo 3. Neste momento, também são explanadas as vantagens de se utilizar esse tipo de treinamento em relação ao treinamento tradicional; as configurações e abordagens de

transferências possíveis; e as principais transferências de aprendizado para redes neurais convolucionais.

A Teoria da Ressonância Adaptativa é apresentada no Capítulo 4. São destacadas suas características que justificam a alternativa ao aprendizado profundo clássico. Adicionalmente, são apresentadas as principais arquiteturas clássicas das redes neurais baseadas nessa teoria, seus embasamentos matemáticos e os algoritmos em fluxogramas.

No Capítulo 5 é apresentada a proposta da tese: duas variantes de modelos híbridos de redes neurais que, pela técnica de transferência de aprendizado, integraram-se o Aprendizado Profundo à Teoria da Ressonância Adaptativa. Além disto, é explanado a motivação da escolha das redes neurais bases para o modelo híbrido e suas principais características.

No Capítulo 6 é realizada a validação por meio de diferentes simulações e seus respectivos resultados. Para comparação aos modelos híbridos propostos, foi criada uma terceira rede neural composta apenas por conceitos de aprendizado profundo. Neste Capítulo também são apresentados diferentes bancos de dados utilizados que visam validar os modelos em múltiplas aplicações de problemas de visão computacional. Além disto, é explorado a vulnerabilidade dos modelos à ataques adversários (introduzidos no Capítulo 1) por meio de testes adversários a partir dos modelos pré-treinados.

Finalmente, no Capítulo 7 são apresentadas e exploradas em diversos contextos as conclusões finais trabalho de pesquisa e as sugestões para trabalhos futuros que podem ser desenvolvidos seguindo esta linha de atuação.

2 REDES NEURAIS CONVOLUCIONAIS

As formas tradicionais de RNAs apresentam grandes limitações para processamento de dados em imagens, pois, normalmente, a imagem é apresentada a partir de um processo de vetorização e, portanto, o número de pixels da imagem se torna correspondente ao de neurônios na camada de entrada da rede neural. Mesmo para imagens com dimensões pequenas a vetorização de imagens exige uma elevada capacidade de memória e poder computacional, principalmente na fase de treinamento.

Se considerar, por exemplo, uma imagem colorida de 64×64 pixels - portanto, com 3 canais - como entrada de uma RNA tradicional, o número de pesos associados a apenas um único neurônio da primeira camada será de $64 \times 64 \times 3 = 2.288$. A vetorização de uma imagem ainda apresenta um problema adicional: despreza toda a estrutura espacial que permite entender a relação entre os pixels vizinhos em uma determinada imagem.

Uma Rede Neural Convolucional - *Convolutional Neural Network* (CNN), por outro lado, é capaz de capturar padrões espaciais por meio da camada convolucional uma vez que, a convolução permite filtrar as imagens considerando sua estrutura bidimensional. Utilizar o processo de convolução também torna o processamento de imagens computacionalmente gerenciável por meio da filtragem das conexões por proximidade.

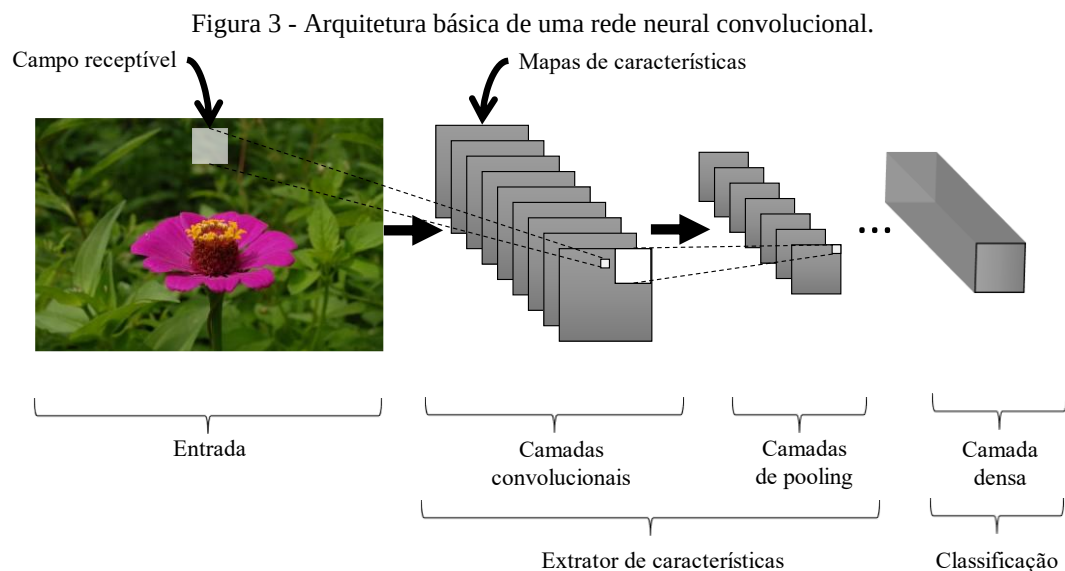
Ao invés de conectar cada pixel a um neurônio, as CNNs restringem as conexões para que um neurônio receba como entrada apenas uma pequena subseção da camada anterior, como, por exemplo, um conjunto de 5×5 pixels. Esta característica permite que cada neurônio seja responsável por processar apenas certa porção de uma imagem, análogo ao funcionamento de neurônios corticais no cérebro, uma vez que cada neurônio responde apenas a uma pequena porção do seu campo visual completo.

Desde sua introdução na década de 1980 (Denker *et al.*, 1988; Fukushima; Miyake, 1982; LeCun, 1989; LeCun *et al.*, 1989), a CNN foi projetada como uma categoria de rede neural especializada em dados visuais. Entretanto, essa arquitetura também é frequentemente utilizada para outros tipos de aplicações, tais como, processamento de linguagem natural (Kalchbrenner; Grefenstette; Blunsom, 2014; Poria *et al.*, 2016; Wang *et al.*, 2015) e reconhecimento de fala (Abdel-Hamid *et al.*, 2012; Huang *et al.*, 2019; Xiong *et al.*, 2018).

2.1 ARQUITETURA DA REDE NEURAL CONVOLUCIONAL

A arquitetura de CNN é composta por uma hierarquia multicamadas de três tipos: camadas convolucionais, camadas de subamostragem (popularmente citadas como camadas de *pooling*) e camadas densas. Na Figura 3 é ilustrado um exemplo de uma arquitetura de

CNN simplificada. Por ser tratar de uma rede neural profunda, este modelo consiste em uma camada de entrada, uma camada de saída e muitas camadas ocultas. As camadas ocultas do modelo CNN consistem, principalmente, em camadas convolucionais, camadas de *pooling* e, em algumas arquiteturas, camadas totalmente conectadas.

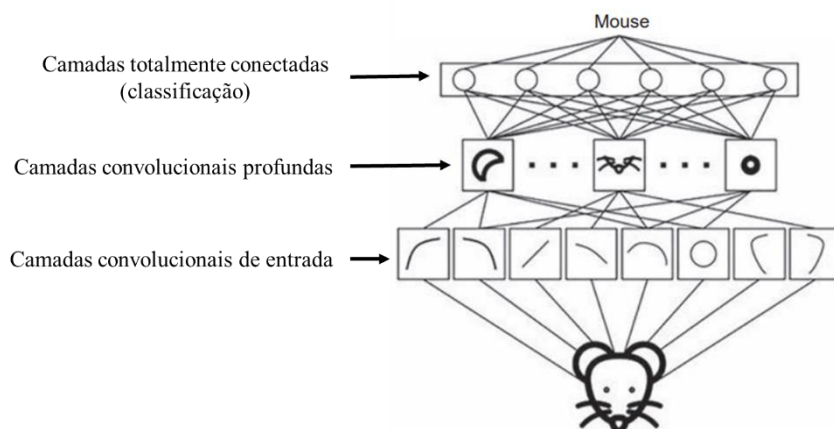


Fonte: Elaboração do próprio autor.

As camadas convolucionais usam um banco de filtros para realizar múltiplas transformações para extrair recursos úteis de informações correlacionadas localmente e armazená-las nos mapas de características (*features maps*). Camadas de *pooling* fazem a subamostragem da informação além de tornar a entrada invariante a distorções geométricas. As camadas densas, também referidas como camadas totalmente conectadas, normalmente utilizadas no final da arquitetura, combinam e processam as características extraídas pelas camadas antecedentes para realizar a classificação.

A quantidade de camadas define a profundidade da rede, do mesmo modo que as redes mais profundas, normalmente, são capazes de identificar padrões com níveis mais altos detalhamentos. O funcionamento geral de uma CNN de várias camadas para classificação de imagens é ilustrado na Figura 4.

Figura 4 - Operação de uma CNN para classificação de imagens.



Fonte: Adaptado de: (Huang; Le, 2021).

As primeiras camadas convolucionais são as responsáveis por extrair características simples da imagem de entrada, como bordas, linhas e assim por diante. Em seguida, os recursos resultantes são enviados para as camadas mais profundas para gerar recursos de maior especificidade, como representações de nariz, olhos e ouvidos. As etapas de extração de características de imagens são aprofundadas nas Seções 2.1.1 e 2.1.2. Finalmente, esses recursos de saída são nivelados em um vetor de recursos unidimensional, em seguida, alimenta as camadas totalmente conectadas (Seção 2.1.3) para classificação/previsão final.

2.1.1 Camada convolucional

Estas camadas realizam convoluções entre a imagem de entrada (ou mapa de características) e os diversos filtros da CNN para extração de características. Um filtro - também conhecido como máscara, núcleo ou *kernel* - é uma matriz com coeficientes que atuam de modo a processar os valores referentes ao nível de cada pixel da imagem original para gerar uma imagem filtrada. Os filtros de uma rede convolucional são equivalentes aos neurônios enquanto seus coeficientes correspondem aos pesos sinápticos da rede e, portanto, são os parâmetros adaptáveis durante à fase de treinamento.

Um filtro pode corrigir, suavizar ou realçar determinadas regiões. O tamanho do filtro os valores de seus coeficientes definem qual o tipo de filtragem é aplicado. Um filtro também pode fornecer uma medida de quão próximo uma região da entrada se assemelha a uma característica - uma *feature*. Esta característica pode ser qualquer aspecto proeminente, *e.g.*, uma aresta vertical, uma aresta horizontal, um arco, uma diagonal etc.

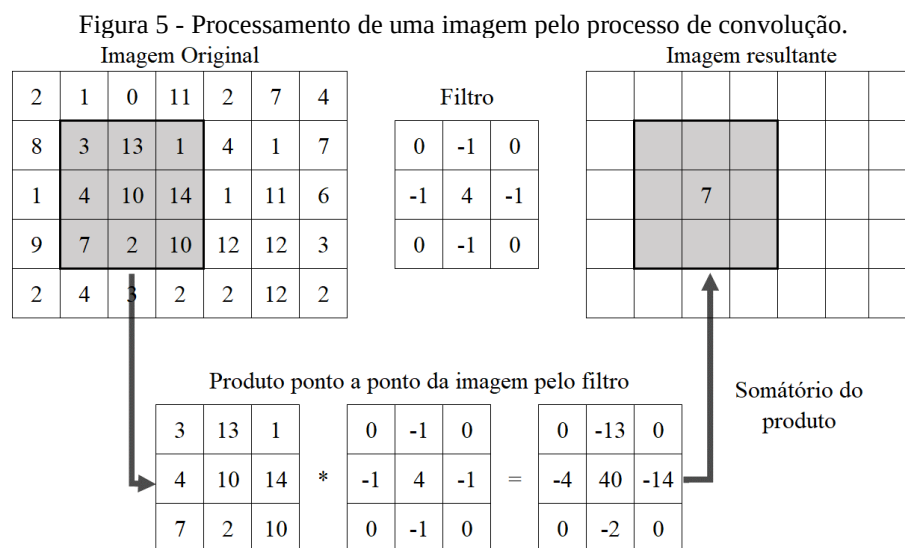
No processo de convolução, o filtro percorre a imagem original “deslizando” sobre os pixels e, a cada interação, sobrepõe um pixel-alvo na imagem e realiza o produto ponto a ponto dos elementos da região sobreposta com os elementos da máscara de posição

equivalente para gerar, no pixel equivalente na imagem filtrada, um valor de intensidade processado. Normalmente, o pixel-alvo é o pixel central da máscara.

Matematicamente, a partir de uma imagem f e uma máscara w , a convolução bidimensional discreta é definida pela Equação (2):

$$w(x, y) * f(x, y) = \sum_{i=-a}^a \sum_{j=-b}^b w(i, j) \cdot f(x - i, y - j) \quad (2)$$

Em que x e y correspondem aos índices horizontais e verticais da imagem $f(x, y)$ e da máscara $w(x, y)$, a e b se referem, respectivamente, à metade da altura e largura da máscara w . Na Figura 5 é apresentado um exemplo de diagrama do processo de convolução (Backes; Sá Junior, 2019).



Fonte: Elaboração do próprio autor.

Por questões de simetria, é comum a utilização de máscaras de tamanho $N \times N$ de modo que N é um número ímpar e, por questão computacional para a maioria das aplicações, recomenda-se utilizar valores pequenos de N ; de modo geral: $N \leq 7$ (Backes; Sá Junior, 2019).

A seguir, serão abordadas algumas propriedades e hiperparâmetros fundamentais para o processamento de *features* em uma camada de convolução. Embora fundamentais, esses hiperparâmetros não são exclusivos da camada convolucional e são comumente utilizados nas camadas de *pooling* (Seção 2.1.2), por exemplo.

2.1.1.1 Mapas de características

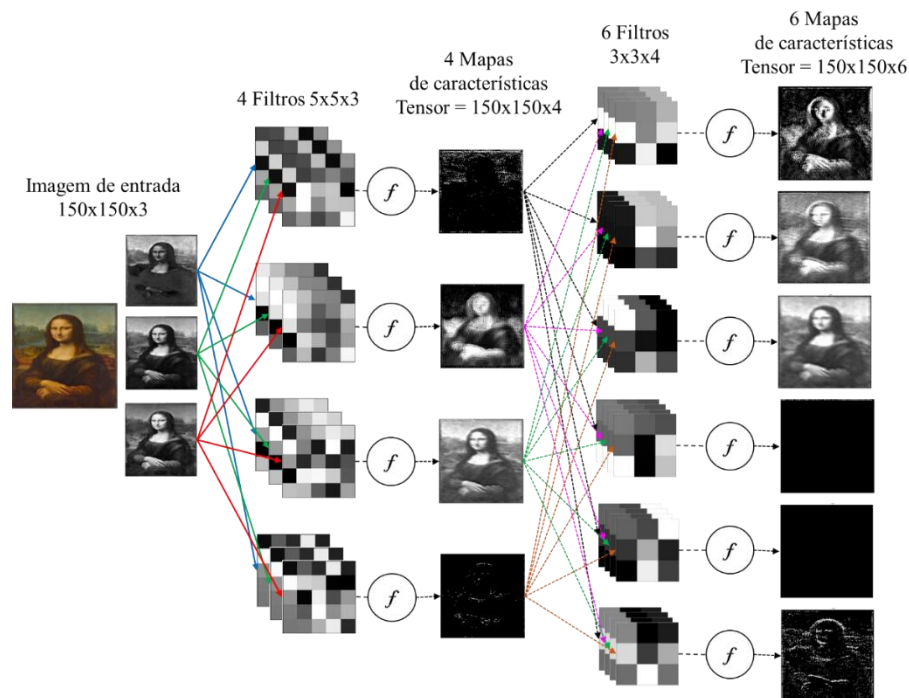
Os mapas de características, ou mapas de ativação (frequentemente abordados por seus termos em inglês: *feature map* ou *activation map*), capturam o resultado da aplicação dos

filtros à entrada, que pode ser a imagem de entrada ou um outro mapa de características. Cada filtro na camada convolucional extrai alguma característica e os diversos mapas gerados são empilhados de modo a formar matrizes multidimensionais, ou tensores, cuja profundidade é igual ao número de filtros.

Entender as dimensões das camadas convolucionais, e o que elas representam, permite entender o comportamento de extração de características da rede convolucional. Seja uma CNN com L camadas convolucionais, $l = 1, 2, \dots, L$. A l – ésima camada é composta por n mapas de características $f_l \in R^{h_l \times w_l \times d_l}$ (altura h_l , largura w_l e profundidade d_l) que são resultados da convolução de n filtros com os mapas de características da camada anterior f_{l-1} . Uma segunda notação - mais intuitiva - representa as dimensões dos mapas de características como $d_l @ h_l \times w_l$.

Na Figura 6 ilustra-se um exemplo de CNN composta por duas camadas convolucionais, com 4 filtros de tamanho $k_1 = 5 \times 5$ e 6 filtros de tamanho $k_2 = 3 \times 3$, respectivamente.

Figura 6 - Ilustração de duas camadas convolucionais.



Fonte: Elaboração do próprio autor.

Na entrada da CNN do exemplo da Figura 6 é apresentada uma imagem RGB, o que justifica a dimensão $150 \times 150 \times 3$. Adicionalmente, considera-se que a convolução é feita a partir da extensão da imagem com preenchimento por zeros de modo que seja possível realizar ao mesmo tempo a filtragem para todos os pixels e preservação do tamanho da imagem. Este procedimento é conhecido por *padding* e será abordado na Seção 2.1.1.3.

Como a imagem possui três canais, na primeira camada convolucional, ter-se-á a composição de um tensor de dimensões $4 \times 5 \times 5 \times 3$, ou seja, quatro filtros de tamanho 5×5 , com profundidade de tamanho três. Após a convolução, utiliza-se uma função de ativação, comumente a função ReLU (Seção 2.1.1.5) que trunca os pixels negativos para zero. Deste modo, a saída da primeira camada é composta por quatro mapas de características de dimensões $150 \times 150 \times 4$.

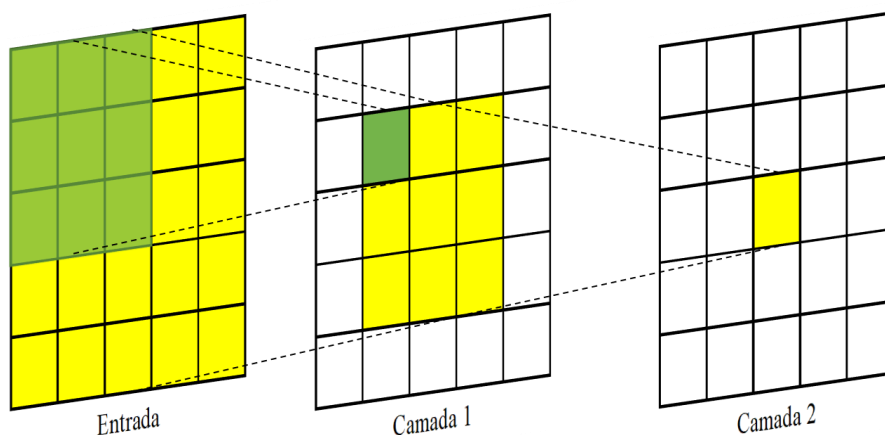
A segunda camada, que recebe como entrada o tensor $150 \times 150 \times 4$, possui seis filtros de dimensões $3 \times 3 \times 4$ e gera como saída um tensor de tamanho $150 \times 150 \times 6$. A partir do exemplo é possível notar que a dimensão do tensor da camada convolucional anterior gera a profundidade da camada seguinte. Portanto, no exemplo supracitado, se fosse adicionado uma terceira camada convolucional, os filtros teriam profundidade seis.

2.1.1.2 Campo receptível

Nas camadas totalmente conectadas, cada unidade tem acesso a toda a região de entrada. Entretanto, em uma camada convolucional cada unidade depende apenas de uma região local da entrada. Para CNNs, o tamanho desta região da entrada, que produz uma *feature*, é definido como Campo Receptivo, ou *Receptive Field* (RF). Em outras palavras, o RF é a região da entrada que uma *feature* particular é afetada.

A Figura 7 ilustra exemplos de campos receptíveis para uma rede de duas camadas convolucionais compostas apenas por filtros de dimensões 3×3 . A área verde na camada de entrada denota o campo receptivo de um pixel na primeira camada convolucional, de mesmo modo, a área em amarelo da primeira camada convolucional representa o campo receptivo de um pixel na segunda camada.

Figura 7 - Exemplos de campos receptíveis.



Fonte: Elaboração do próprio autor.

O principal interesse no tamanho do campo receptivo concerne à entrada; especificamente, ao quanto de sua área a CNN é capaz de processar. A área da entrada que ativa um neurônio nas camadas convolucionais subsequentes é denominada de Campo Receptivo Efetivo, ou *Effective Receptive Field* (ERF): Ainda no exemplo da Figura 7, na camada de entrada, a área em amarelo representa o ERF da *feature* destacada na segunda camada.

O ERF é essencial em muitas tarefas de visão computacional, como por exemplo, a detecção de objetos e segmentação de imagens, uma vez que, se a rede neural convolucional não tiver um campo receptivo grande o suficiente, alguns dos objetos maiores na imagem de entrada podem não ser detectados e/ou segmentados. Calcular o tamanho do ERF pode ajudar a projetar os tamanhos de filtros adequados para melhorar o desempenho das CNNs (Le; Borji, 2017).

O tamanho do RF é simplesmente igual ao do filtro da camada anterior e, portanto, RF e ERF serão os mesmos na primeira camada convolucional. No entanto, conforme a arquitetura da CNN avança com as camadas de convolução empilhadas, os campos receptíveis apresentam uma relação piramidal.

A partir de uma rede totalmente convolucional, ou *Fully Convolutional Network* (FCN) - ou seja, uma rede sem camadas totalmente conectadas (abordado na seção 2.1.3) - com L camadas, $l = 1, 2, \dots, L$. O ERF de uma imagem na camada de entrada pode ser obtido de forma recursiva pela Equação (3) (Araujo; Norris; Sim, 2019):

$$r_0 = \sum_{l=1}^L \left((k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1 \quad (3)$$

Tal que:

r_0 : Tamanho do campo receptível em relação à entrada

k_l : Tamanho do filtro da l - ésima camada

s_i : Tamanho do *stride* (Seção 2.1.1.4) das camadas anteriores a l .

2.1.1.3 Padding

No processo de convolução, as bordas da imagem precisam receber um tratamento especial, uma vez que parte do filtro sobrepõe pixels inexistentes. O *padding* é o tratamento mais popular caso seja necessário manter as informações apresentadas nas bordas e a dimensionalidade na imagem resultante durante a operação de convolução. Este procedimento consiste em assumir o valor zero para os pixels ao redor da imagem, conforme ilustrado na Figura 8.

convolução. A Equação (4) pode ser utilizada para calcular as dimensões, tanto na horizontal quanto na vertical, de saída.

$$n_{out} = \frac{n_{in} + 2p - k}{s} + 1 \quad (4)$$

Tal que:

- n_{out} : dimensão da saída;
- n_{in} : dimensão da entrada;
- p : dimensão do *padding*;
- k : dimensão do filtro;
- s : dimensão do *stride*.

A redução, nas dimensões dos mapas de características, produz pouca invariância à escala e à posição dos objetos, o que é uma propriedade útil em tarefas de reconhecimento de objetos. Entretanto, em algumas aplicações, como redução de ruído ou segmentação de imagem, é importante manter as dimensões espaciais constantes (ou até maiores) após a convolução, visto que estas aplicações exigem previsões com maior dependência ao nível de pixel. Além disso, o controle da dimensão permite projetar redes mais profundas (ou seja, com mais camadas de peso), evitando um colapso rápido das dimensões do recurso de saída.

2.1.1.5 Função de ativação

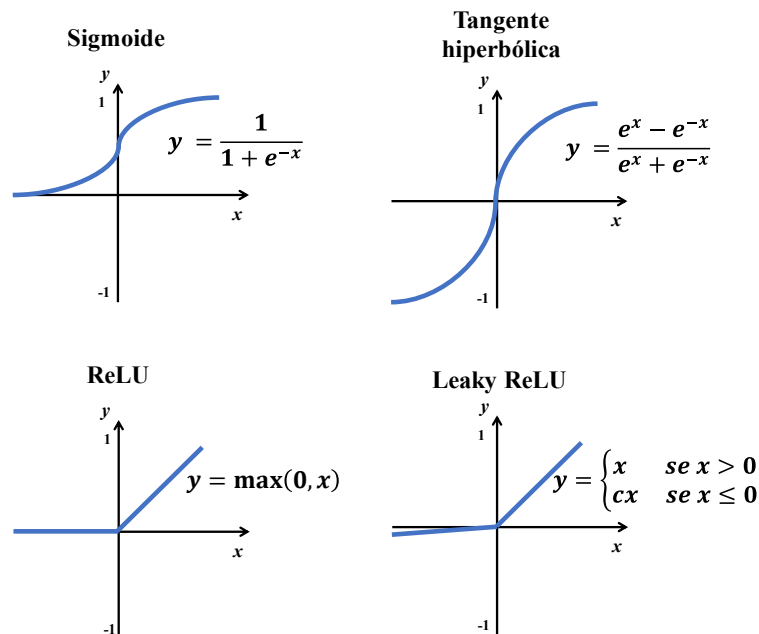
A operação de convolução ajuda no processo de extração de recursos úteis em pontos de dados correlacionados localmente e os valores das saídas dos filtros convolucionais, uma entrada de valor real, é então atribuída à unidade de processamento não linear (função de ativação), que restringe para este valor um pequeno intervalo, como $[0; 1]$ e $[-1; 1]$.

A aplicação de uma função de ativação após as camadas de peso é muito importante, que não apenas ajuda no aprendizado de abstrações, mas também incorpora a não linearidade no espaço de recursos (Goodfellow; Bengio; Courville, 2016). Uma rede neural sem função de ativação torna-se um modelo linear. Na ausência de não linearidades, uma rede empilhada de camadas de peso é equivalente a um mapeamento linear do domínio de entrada para o domínio de saída.

Uma função não linear também pode ser entendida como um mecanismo de comutação ou seleção, que decide se um neurônio irá disparar ou não, dadas todas as suas entradas. Por causa do algoritmo de retropropagação, comumente utilizado nas redes profundas, baseado em gradiente descendente, as funções de ativação devem ser diferenciáveis para permitir a propagação de erro. Algumas das funções de ativação mais

comuns que são usadas em redes neurais profundas são ilustradas na Figura 10 (Li *et al.*, 2022).

Figura 10 - Exemplos de funções de ativações usadas em redes neurais profundas.



Fonte: Elaboração do próprio autor.

No contexto de imagens, a função de ativação mais popular nas camadas convolucionais é a função ReLU (*Rectified Linear Activation Function*) que cancela todos os valores negativos e possui um comportamento linear para os positivos. A ReLU apresenta comportamento simples o que permite seu rápido processamento em cálculo, característica com uma importância prática especial em redes neurais profundas. A popularidade e a eficácia da ReLU fomentaram o desenvolvimento de outras variantes que abordam algumas das deficiências da função de ativação ReLU, por exemplo, ReLU com vazamento (ou *Leaky ReLU*) que não reduz completamente as entradas negativas para zero (Dubey; Jain, 2019; Khan *et al.*, 2018; Li *et al.*, 2022; Nair; Hinton, 2010).

2.1.2 Camada de pooling

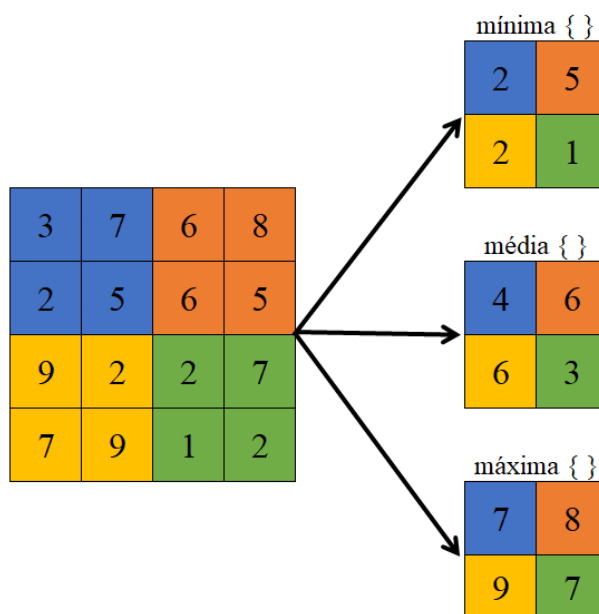
A saída da função de ativação não linear geralmente é seguida por uma subamostragem (*pooling*) que ajuda a resumir os resultados e aumenta a robustez da entrada contra distorções geométricas como mudanças de escala, posicionamento e translação. Ou seja, a camada de *pooling* recebe como entrada os valores dos mapas de características processados com a não linearidade e combina seus coeficientes por meio de uma função de agrupamento. As funções de *pooling* mais populares são de média ou máxima - popularmente

citadas como *average pooling* ou *max pooling* (Goodfellow; Bengio; Courville, 2016; Khan *et al.*, 2018).

Como resultado, se tem uma matriz agrupada com apenas partes dos dados de entrada - os mais significativos – e, conseqüentemente, os cálculos são minimizados; além de evitar o problema de superposição de informação.

Semelhante à camada de convolução, na camada de *pooling* se faz necessário configurar o tamanho da região agrupada e o passo. Na Figura 11 são ilustrados exemplos de operações de *pooling* 2×2 de mínima, média e máxima, ambos com o *stride* 2.

Figura 11 - Operações de *pooling*: tamanho 2×2 e *stride* 2.



Fonte: Elaboração do próprio autor.

2.1.3 Camada totalmente conectada

As matrizes multidimensionais, que representam a saída da camada convolucional, passam por um processo de vetorização, ou achatamento (*flatten*), na camada totalmente conectada - ou *Fully Connected* (FC). A camada FC corresponde essencialmente a camadas de convolução com filtros de tamanho 1×1 e cada unidade em uma camada totalmente conectada é densamente conectada a todas as unidades da camada anterior.

Por exemplo, se a camada convolucional anterior à camada FC propaga um tensor de dimensão $4 \times 4 \times 5$, esses dados são “achutados” para um vetor de dimensão $1 \times (4 \times 4 \times 5) = 1 \times 80$. Portanto, cada neurônio na camada FC possuirá 80 pesos de forma a produzir uma combinação linear do vetor. De fato, uma camada totalmente conectada é idêntica a uma camada de peso da arquitetura *Multi-layer Perceptron*.

Em muitos casos, as camadas totalmente conectadas são posicionadas apenas no final da arquitetura. No entanto, algumas arquiteturas de redes convolucionais utilizam esse tipo de camada em um posicionamento intermediário (Khan *et al.*, 2018).

2.2 ARQUITETURAS POPULARES DE CNNs

Neste tópico são apresentadas as arquiteturas mais populares, consideradas a “espinha dorsal” das CNNs. Dentre elas, serão dispostas tanto as arquiteturas pioneiras, populares em visão computacional e de simples entendimento, e alguns dos modelos mais recentes da CNN, relativamente mais complexos e construídos tendo como base os projetos convencionais. Deste modo, é possível notar que há uma ascensão natural de técnicas e recursos nessas arquiteturas, evidenciando a evolução dos projetos ao longo dos últimos anos.

Muitos dos modelos abordados a seguir participaram do *ImageNet LargeScale Visual Recognition Challenge* (ILSVRC), uma competição que ocorreu entre os anos de 2010 e 2017 e tornou-se referência em desafios de detecção e classificação de objetos em imagens.

O principal pretexto do seu sucesso é a utilização de um conjunto de dados gigantesco: o ImageNet; que, a partir da versão de 2012 é composto por mais de 15 milhões de imagens de alta resolução rotuladas em cerca de 22.000 categorias. O ILSVRC usa um subconjunto de 1.000 categorias, sendo cada uma com, aproximadamente, 1.000 imagens. Ao todo, 1,28 milhões de imagens são utilizadas para o treinamento, mais 50.000 imagens para validação e 100.000 imagens para o teste (Deng *et al.*, 2009; Russakovsky *et al.*, 2015b).

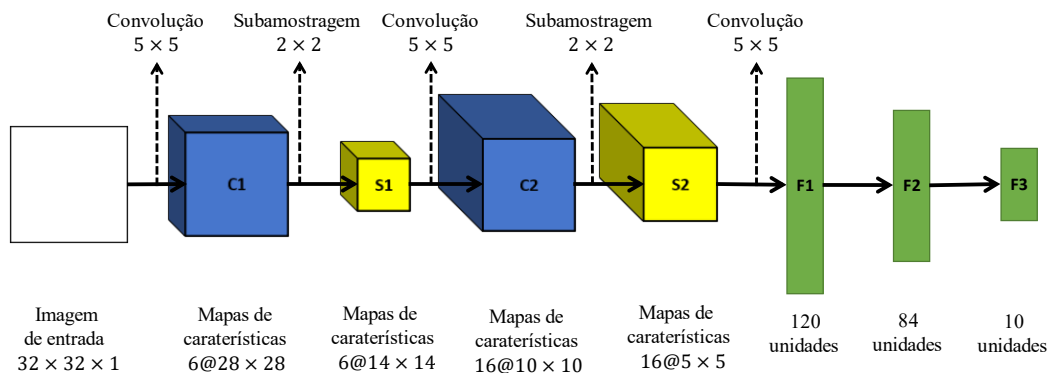
2.2.1 LeNet

Uma das arquiteturas pioneiras de CNNs é a LeNet-5, aplicada à identificação de dígitos manuscritos. Este modelo é muito popular por apresentar um comportamento simples e direto. Seu nome se refere ao autor e à sua arquitetura, composta por um total de 5 camadas de pesos (LeCun *et al.*, 1998).

Na devida ordem, a LeNet-5 - Figura 12 - consiste em: uma camada de entrada que recebe imagens de 32×32 pixels; uma camada convolucional composta por 6 filtros de dimensões 5×5 seguida por uma camada de subamostragem 2×2 para extrair recursos; uma segunda camada de convolução composta por 16 filtros de 5×5 seguida por mais uma camada de subamostragem 2×2 ; posteriormente, uma camada de convolução composta por filtros de dimensões 5×5 formam 120 unidades de dimensões 1×1 , seguida por um conjunto de duas camadas totalmente conectadas, com respectivamente, 84 e 10 neurônios, no final do modelo para atuar como classificador dos recursos extraídos. As ativações após as

camadas de pesos são comprimidas por não linearidades do tipo sigmoide, tangente hiperbólica e a função de base radial.

Figura 12 - Arquitetura LeNet-5.



Fonte: Adaptado de: (LeCun *et al.*, 1998).

2.2.2 AlexNet

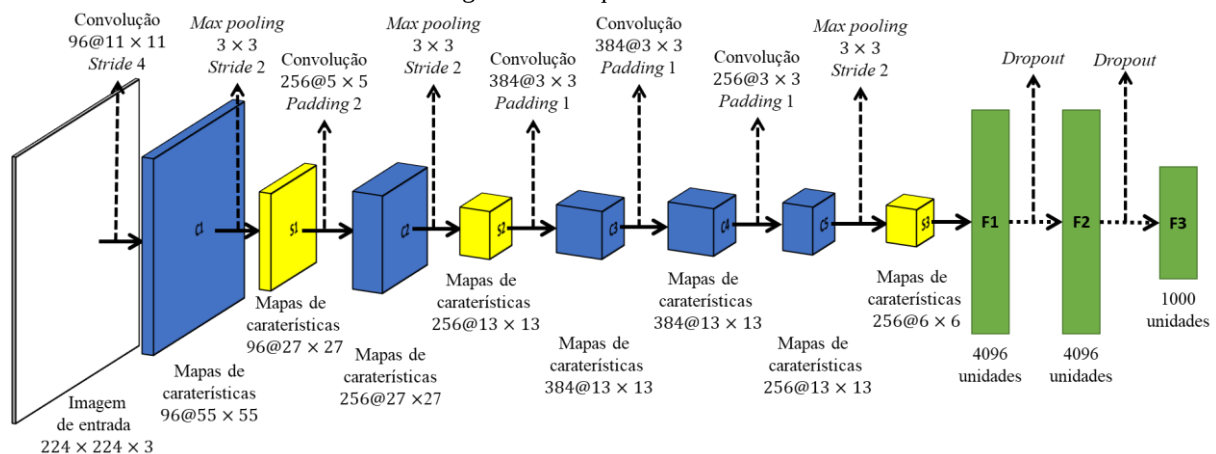
Vencedora da ILSVRC 2012, a rede neural AlexNet foi a primeira arquitetura de CNN a ser utilizada em larga escala e levou ao ressurgimento de redes neurais profundas em visão computacional (Krizhevsky; Sutskever; Hinton, 2012).

A principal diferença entre a arquitetura AlexNet e seus antecessores é o aumento da profundidade da rede, que leva a um número significativamente maior de parâmetros ajustáveis, além de truques de regularização usados, tais como, o *dropout* de ativação e o *data augmentation*, o que proporciona a redução do *over-fitting* e uma melhor generalização para exemplos não vistos (Hinton *et al.*, 2012; O'Shea; Nash, 2015; Srivastava *et al.*, 2014).

Outro aspecto diferenciador do AlexNet é o uso da não linearidade ReLU após cada camada convolucional e totalmente conectada, o que melhora substancialmente a eficiência do treinamento em comparação com as funções sigmoide e tangente hiperbólica (tradicionalmente usadas até aquele momento) que devido às suas naturezas de saturação, sofrem com o problema do *Vanishing Gradient* (VG) e dificultam o treinamento da rede.

Na Figura 13 são representados os detalhes estruturais de cada camada do modelo AlexNet, que consiste em cinco camadas convolucionais, algumas das quais são seguidas por camadas de subamostragem do tipo *max pooling*, e três camadas totalmente conectadas. A rede neural contabiliza mais de 60 milhões de parâmetros de treinamento e 650.000 neurônios.

Figura 13 - Arquitetura AlexNet.



Fonte: Adaptado de: Krizhevsky; Sutskever; Hinton (2012).

A primeira camada convolucional extrai as características da imagem de entrada - RGB de dimensões $224 \times 224 \times 3$ - com 96 filtros de tamanho $11 \times 11 \times 3$ e *stride* de 4 pixels. Em seguida, uma subamostragem de tamanho 3×3 e *stride* 2 os reduz para a dimensão 27×27 . A segunda camada convolucional recebe como entrada a saída normalizada e agrupada da primeira camada convolucional, realiza *padding* 2 e a processa a partir de 256 filtros de tamanho $5 \times 5 \times 96$ com *stride* 1; em seguida, uma nova subamostragem 3×3 e *stride* 2 disponibiliza para a terceira camada convolucional 256 mapas de ativação de dimensões 13×13 .

A terceira, quarta e quinta camadas convolucionais são conectadas umas às outras sem quaisquer camadas de subamostragem ou normalização, entretanto, mantêm as dimensões dos mapas de características devido ao tamanho do filtro (3×3) e a utilização de *padding* de dimensão 1. Isto posto, a terceira camada convolucional dispõe 384 filtros de tamanho $3 \times 3 \times 256$, a quarta camada convolucional de 384 filtros de tamanho $3 \times 3 \times 384$, a quinta camada convolucional de 256 filtros de tamanho $3 \times 3 \times 384$ e, finalmente, uma última subamostragem por *max pooling* 3×3 é realizada antes da etapa de classificação.

Na etapa de classificação, as duas primeiras camadas totalmente conectadas possuem 4096 neurônios cada e são interconectadas pela técnica de *dropout*. A camada final totalmente conectada, ou seja, a camada de saída contém 1.000 unidades, pois, utiliza a função de ativação *softmax* para classificar, a partir dos recursos extraídos da imagem de entrada, em uma das mil classes do conjunto de dados *ImageNet*.

2.2.3 Network in Network

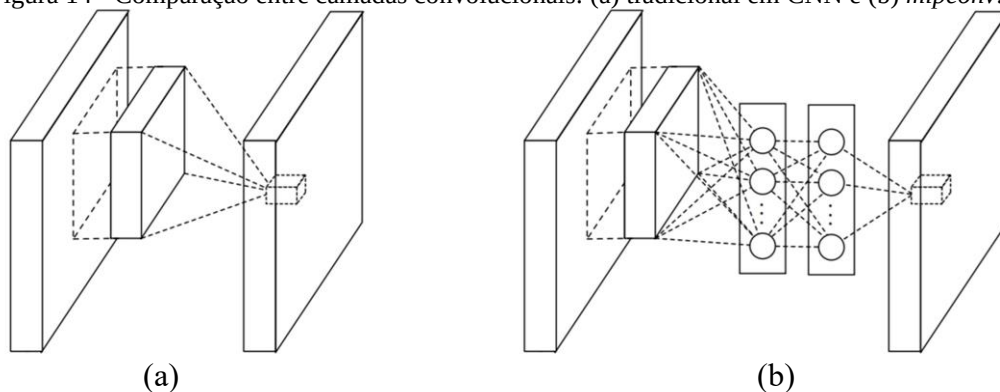
A LeNet, AlexNet e VGG (esta última que será apresentada a seguir) compartilham um padrão de *design* comum: extrair recursos explorando a estrutura espacial por meio de

uma sequência de convoluções e camadas de *pooling* e pós processá-los por meio de camadas totalmente conectadas. As melhorias no LeNet por AlexNet e VGG residem principalmente em como estas redes ampliam e aprofundam esses dois módulos: extração de recursos e classificação.

Estes três modelos apresentam dois grandes desafios: o número de parâmetros exigidos nas camadas totalmente conectadas no final das arquiteturas e a incapacidade de adicionar camadas totalmente conectadas no entremeio da rede para aumentar o grau de não linearidade: este processo destruiria a estrutura espacial e, potencialmente, exigiria mais memória durante o treinamento (Zhang *et al.*, 2021).

Lin; Chen; Yan (2013), autores da arquitetura *Network in Network* (NiN), introduziram duas novas ideias para oferecer uma alternativa capaz de resolver ambos os problemas: Primeiramente, em substituição aos filtros lineares convencionais, propuseram a incorporação de camadas totalmente conectadas do tipo MLP entre as saídas das camadas convolucionais tradicionais, esta estrutura é denominada de camada *mlpconv*. Os neurônios das camadas MLP é equivalente a filtros convolucionais de tamanho 1×1 . Os autores defendem que substituir um filtro convolucional por um aproximador de função não linear mais potente pode aumentar a capacidade de abstração do modelo local e, portanto, aumenta a capacidade de expressividade não linear do modelo. Na Figura 14 é apresentada uma comparação desta estrutura com a de uma camada convolucional tradicional.

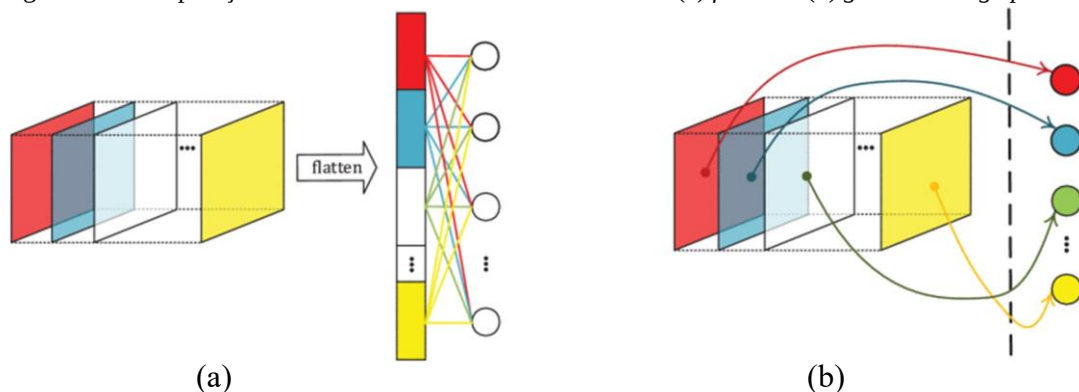
Figura 14 - Comparação entre camadas convolucionais: (a) tradicional em CNN e (b) *mlpconv*.



Fonte: (Lin; Chen; Yan, 2013).

Outra contribuição da proposta da NiN é a substituição de camadas totalmente conectadas tradicionais vista no final de outros modelos por subamostragens do tipo *pooling* médio global. Conforme comparação representada na Figura 15, este esquema de *pooling* combina, por média, todas as ativações dentro de cada mapa de características para obter uma única pontuação de classificação que é, em seguida, encaminhada para uma camada do tipo *soft-max* para classificação.

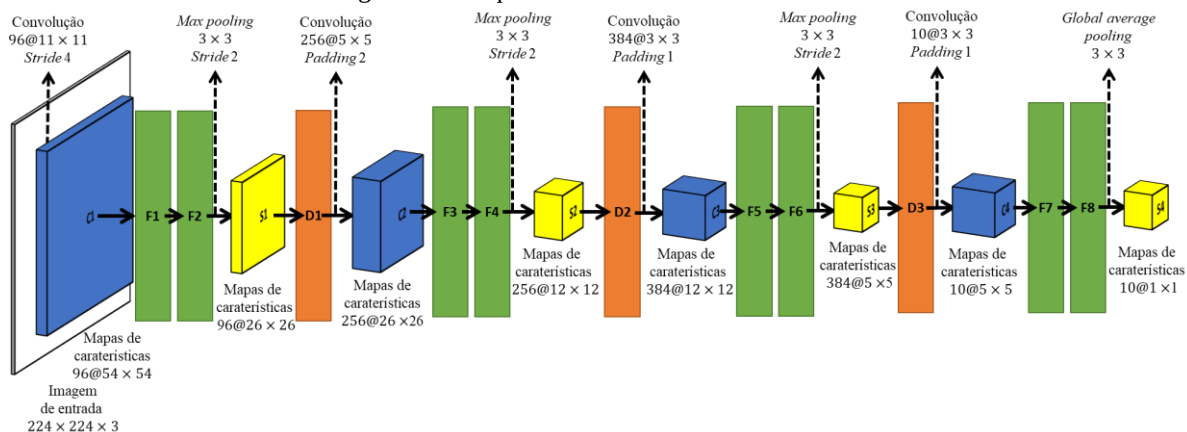
Figura 15 - Comparação entre camadas totalmente conectadas: (a) *flatten* e (b) *global average pooling*.



Fonte: (Bu *et al.*, 2020).

A proposta deste modelo de CNN é diminuir parâmetros com uma rede neural simples e leve e, deste modo, sua principal aplicação se dá em conjuntos de dados de pequena escala em dispositivos móveis e sistema embarcados (Khan *et al.*, 2018).

Figura 16 - Arquitetura da *Network in Network*.



Fonte: Adaptado de: Khan *et al.* (2018); Lin; Chen; Yan (2013); Zhang *et al.* (2021).

A arquitetura da NiN, ilustrada na Figura 16, procede das mesmas estruturas iniciais de convolução do AlexNet, *i.e.*, consiste em camadas convolucionais com filtros de tamanho 11×11 , 5×5 e 3×3 , respectivamente. Cada camada convolucional é seguida por um par de camadas totalmente conectadas e uma camada de *max pooling* com janela de 3×3 e *stride* de tamanho 2. Blocos de regularização *dropout* são utilizados após todas as camadas de *max pooling* - com exceção da última - para contribuir em direção a um menor erro de teste e maior generalização da rede. Como supracitado, na última camada *mlpconv*, utiliza-se uma subamostragem do tipo *pooling* médio global (*Global average pooling*) para extrair a média de cada mapa de característica e gerar um vetor resultante propagado a uma camada classificadora com função do tipo *softmax*.

2.2.4 VGGnet

Embora não tenha sido o vencedor do ILSVRC no ano de 2014, a arquitetura VGG (Simonyan; Zisserman, 2014) é um dos modelos de CNN mais populares desde a sua introdução em 2014. A razão de sua popularidade está na simplicidade de seu modelo: a proposta de utilizar pequenas dimensões de largura e altura das camadas convolucionais, enquanto expande a profundidade ao decorrer da rede. Com esta proposta, se atinge redes muito profundas com filtros convolucionais pequenos que são capazes de extrair maior detalhamento da entrada.

Seu nome é um acrônimo do departamento de Ciências da Engenharia da Universidade de Oxford *Visual Geometry Group* (VGG). Os autores da VGG introduziram variações de configurações de VGGnets, que seguem o mesmo projeto genérico composto em blocos, com a diferença apenas nas profundidades e, de acordo com eles, a principal contribuição de seu trabalho é uma avaliação completa de redes de profundidade crescente.

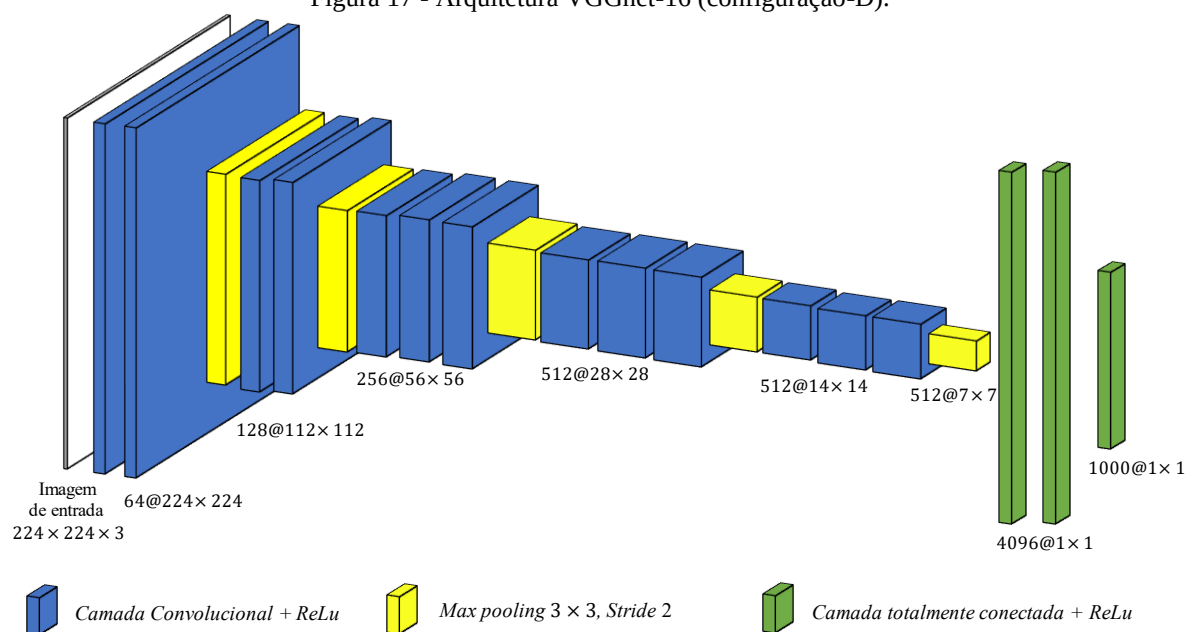
A configuração de VGGnet menos profunda (a configuração A) possui 11 camadas de peso, sendo 8 camadas convolucionais e 3 camadas FC; à medida que, a mais profunda (configuração E) possui 19 camadas de peso na rede, sendo 16 camadas convolucionais e 3 camadas FC. Dentre todas as variações, as configurações D e E (comumente referidas como VGGnet-16 e VGGnet-19 na literatura) são as mais bem sucedidas.

Os blocos do projeto genérico das VGGnets são constituídos a partir da sequência: uma camada convolucional com *stride* 1 e *padding*, para manter a resolução, uma não linearidade como uma ReLU e uma camada de *max-pooling* com janela de 2×2 e *stride* 2 para reduzir a resolução. Com exceção de apenas uma configuração que utiliza filtros de convolução 1×1 , todas as demais arquiteturas VGGnet são constituídas estritamente de filtros de convolução de tamanho 3×3 - o menor tamanho necessário para capturar a noção de pixels vizinhos.

O conjunto de camadas convolucionais (que tem uma profundidade diferente em diferentes arquiteturas) é seguido por três camadas totalmente conectadas: as duas primeiras têm 4096 unidades cada e a terceira e final, que é composta pela ativação *soft-max* para realizar classificação da ILSVRC, contém, portanto, 1000 unidades. A configuração das camadas totalmente conectadas é a mesma para todas as VGGnets.

Na Figura 17 têm-se representado o modelo VGGnet-16. Este modelo possui 138 milhões de parâmetros treináveis e, semelhante ao AlexNet, usa *dropouts* de ativação nas duas primeiras camadas totalmente conectadas para evitar o *overfitting*.

Figura 17 - Arquitetura VGGnet-16 (configuração-D).



Fonte: Adaptado de: (Simonyan; Zisserman, 2014).

Os autores apontam que existem duas vantagens na estratégia de utilização de filtros 3×3 empilhados em relação a uma única camada de filtros de maior dimensão. A primeira é triplicar o número de retificação não lineares, o que torna a função de decisão mais discriminativa.

Em segundo lugar, a diminuição do número de parâmetros mantendo a dimensão do campo receptivo efetivo. Visto que, por exemplo, o empilhamento de duas camadas convolucionais com filtros de tamanho 3×3 (sem subamostragem espacial entre elas) proporciona um campo receptivo efetivo equivalente ao de uma camada composto por filtros 5×5 ; três destas camadas dispõem equivalência a um campo receptivo efetivo de uma camada com filtros 7×7 . Ou seja, supondo uma pilha de três camadas de convolução 3×3 tenham C canais de profundidade, a pilha é parametrizada por $3(3^2C^2) = 27C^2$ pesos; por outro lado, uma camada convolutiva de 7×7 exigiria $1(7^2C^2) = 49C^2$ pesos, ou seja, 81% parâmetros a mais (Simonyan; Zisserman, 2014).

Deste modo, filtros menores reduzem relativamente o número de parâmetros e, portanto, proporcionam treinamentos e testes mais eficientes e, conseqüentemente, permitem empilhar mais camadas que oportunizam redes mais profundas com melhores desempenhos nas tarefas de visão computacional.

2.2.5 GoogLeNet

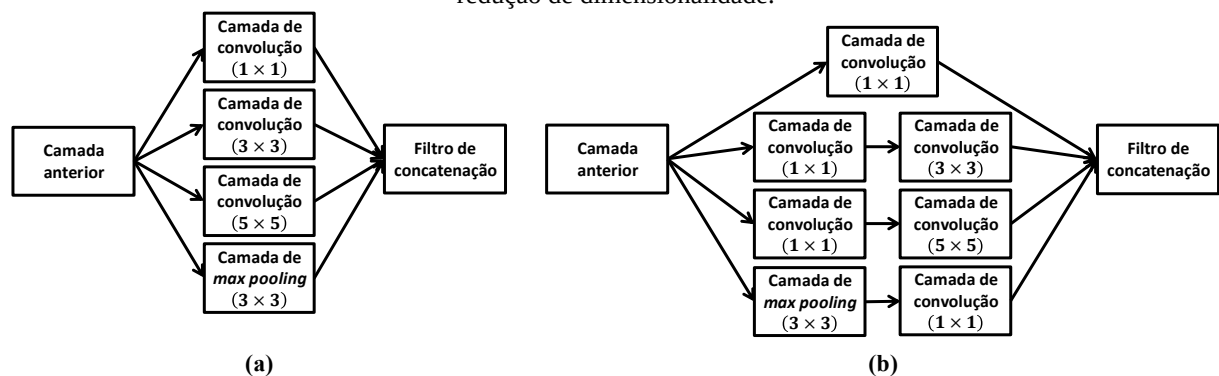
As redes supracitadas consistem em uma arquitetura sequencial com camadas empilhadas umas sobre as outras para criar uma arquitetura profunda composta por apenas um

caminho. Neste caminho, diversas camadas podem ser utilizadas, tais como: convolução, *pooling*, ReLU, *dropout* e camadas totalmente conectadas.

A arquitetura GoogLeNet (Szegedy *et al.*, 2015) é o primeiro modelo popular que utiliza uma arquitetura mais complexa com várias ramificações. Este modelo, vencedor da competição ILSVRC no ano de 2014, usa em sua estrutura os pontos fortes do NiN (Lin; Chen; Yan, 2013), combinados a blocos repetidos (Simonyan; Zisserman, 2014) e uma composição de filtros convolucionais. Várias versões aprimoradas e estendidas do GoogLeNet foram propostas a posteriori, no entanto, nesta seção, apenas a arquitetura enviada ao ILSVRC 2014 do GoogLeNet será discutida.

O bloco fundamental de construção da rede GoogLeNet é o módulo “*Inception*”, representado na Figura 18. Este módulo ganhou tanto destaque que a arquitetura da rede é também comumente mencionada como “*Inception Network*”.

Figura 18 - Estrutura do Módulo *Inception*: (a) Conceito básico do módulo *Inception* (b) Módulo *Inception* com redução de dimensionalidade.



Fonte: Adaptado de: (Szegedy *et al.*, 2015).

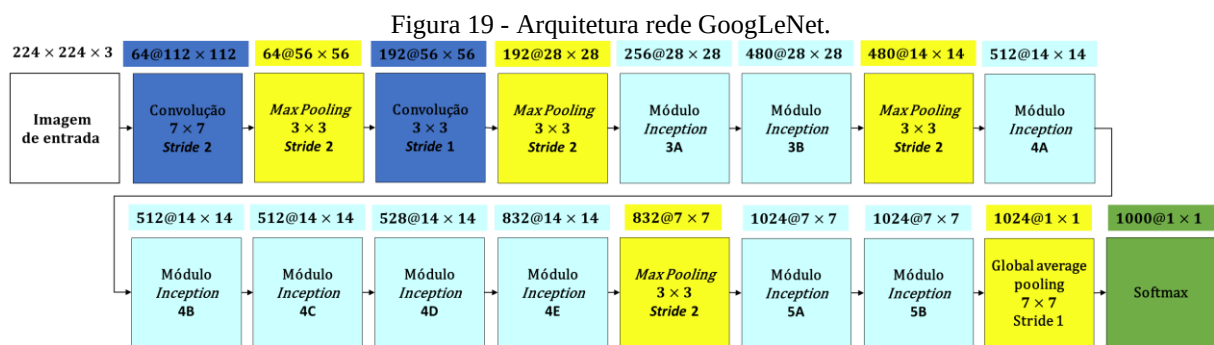
Conforme ilustrado na Figura 18(a), o módulo consiste em quatro ramificações paralelas. As três primeiras ramificações usam, respectivamente, camadas convolucionais com janelas de 1×1 , 3×3 e 5×5 para extrair informações de diferentes tamanhos espaciais. Além disto, de modo que as operações de *pooling* são essenciais para o sucesso das redes convolucionais tradicionais, um caminho alternativo de *pooling* foi adicionado em paralelo a estes estágios. Este caminho reduz as dimensões de altura e largura das entradas, sintetizando as informações da camada anterior, de modo a preservar características que são propagadas para as camadas subsecutivas.

A ideia central do módulo *Inception* é colocar todos os blocos básicos de processamento que ocorrem em uma rede convolucional de forma sequencial regular em paralelo e combinar suas representações na saída. As ramificações usam *padding* e *stride* apropriados para fornecer as mesmas dimensões de altura e largura à saída.

Embora a arquitetura ilustrada na Figura 18 (a) possa cobrir a estrutura esparsa ideal, o problema, no entanto, é que, se todos os mapas de características de cada bloco forem simplesmente concatenados sem nenhuma alteração ao longo da dimensão de profundidade, resultará em um mapa de características de saída de dimensão muito alta, que pode acarretar à uma explosão computacional de processamento em poucos estágios.

Para superar este problema, foi proposto o módulo *Inception* representado na Figura 18 (b), que realiza redução da dimensão da profundidade a partir de convolução de janelas 1×1 antes de passar pela extração de características pelos filtros de convolução 3×3 e 5×5 . Isto significa que, enquanto os filtros de convolução e *pooling* operam no domínio espacial, *i.e.*, ao longo da altura e largura, a camada de dimensões 1×1 combina informações ao longo da profundidade.

A arquitetura GoogLeNet representada na Figura 19 possui 9 módulos *Inceptions* empilhados, que resulta em uma rede profunda de 22 camadas (se considerar as camadas dos módulos, o número total de camadas na rede é maior do que 100). Semelhante ao NiN, o GoogLeNet usa o *pooling* médio global seguido por uma camada totalmente conectada no final da rede de classificação pela função *Softmax*. Para melhorar a capacidade de generalização da rede utiliza-se uma camada de *dropout* antes da camada de classificação. Todas as convoluções, incluindo as dentro dos módulos *Inception*, usam ativação linear retificada. Maiores detalhes da rede, com dimensões exatas de todas as camadas em cada bloco, são disponibilizados em Szegedy *et al.* (2015).



Fonte: Adaptado de: (Szegedy *et al.*, 2015).

Embora a arquitetura do GoogLeNet pareça muito mais complexa do que seus antecessores, como a AlexNet e a VGGnet, ela envolve um número significativamente reduzido de parâmetros (6 milhões, em comparação com 62 milhões no AlexNet e 138 milhões de parâmetros no VGGnet), portanto, apresenta um consumo de memória muito menor, melhor eficiência e alta precisão (Khan *et al.*, 2018).

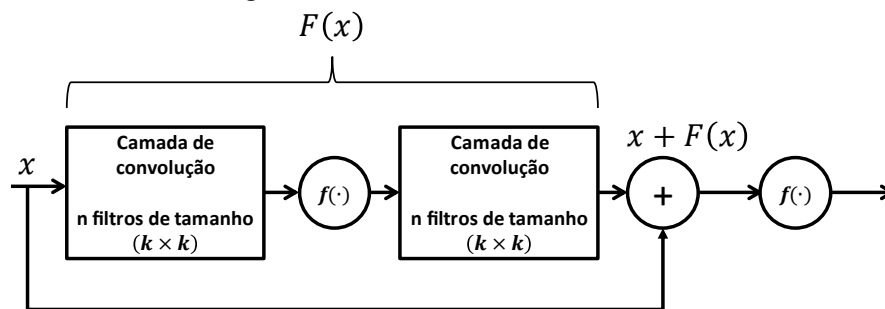
2.2.6 ResNet

A Rede Residual, do inglês *Residual Network* (ResNet), da Microsoft, venceu o desafio ILSVRC 2015 com um grande salto no desempenho: reduziu a taxa de erro *top-5* para 3,6%. A título de comparação, a GoogleLeNet, vencedora do ano anterior, apresentou um desempenho de 6,7%.

A principal característica da arquitetura da ResNet é o Bloco Residual, ilustrado na Figura 20, desenvolvido com uma conexão direta, chamada de “*skip*”, que ignora o treinamento de algumas camadas e se conecta diretamente à saída. Desta forma, o bloco residual combate o problema de convergência, causado por fuga/explosão de gradiente, e possibilita o treinamento de redes substancialmente mais profundas do que as usadas anteriormente.

Esta arquitetura de bloco é semelhante à apresentada na rede *Highway* que apresenta conexões de *skip* com comutação para controlar o fluxo de sinal através de suas ramificações - principal e *skip*. Estes comutadores possuem parâmetros, são dependentes de dados e permite que a rede decida qual ramificação é mais útil para a tarefa final. Em contrapartida, o *skip* proposto na ResNet não é disposto com um parâmetro de treinamento (He *et al.*, 2016; Srivastava; Greff; Schmidhuber, 2015).

Figura 20 - Bloco residual de duas camadas.



Fonte: Adaptado de: (He *et al.*, 2016).

Para entender estas conexões, considere o bloco residual ilustrado na Figura 20: dada uma entrada x , as camadas de pesos CNN juntamente com uma função ReLU implementam uma transformação nesta entrada, representada por $F(x)$. Por intermédio da conexão de *skip*, a entrada é adicionada diretamente a esta transformação. Desta forma, a função de transformação em um bloco residual é dividida em um termo de identidade (que representa a entrada) e um termo residual que ajuda a focar na transformação dos mapas de características de resíduos (He *et al.*, 2016; Khan *et al.*, 2018).

Dado que na saída de um bloco residual tem-se $H(x) = F(x) + x$, as camadas de peso aprendem apenas as funções de representação residual $F(x) = H(x) - x$ em vez de aprender

a representação do sinal diretamente. À medida que o mapeamento de recursos residuais é muitas vezes mais simples do que o mapeamento aprendido nas arquiteturas convencionais, a ResNet alcança um aprendizado estável de modelos muito profundos.

Várias configurações de ResNet foram propostas, na Tabela 1 são apresentadas as configurações de arquiteturas de ResNets desenvolvidas para a ImageNet, publicadas por HE *et al.* (2016). Da mesma forma que o módulo inicial no GoogleNet, a ResNet compreende vários blocos residuais empilhados uns sobre os outros. Todavia, estes blocos se derivam do *design* da camada convolucional 3×3 da VGGnet.

Tabela 1 - Arquiteturas de ResNet para ImageNet.

Nome da camada	Tamanho da saída	18 camadas	34 camadas	50 camadas	101 camadas	152 camadas
conv_1	112×112	$7 \times 7, 64, \text{stride } 2$				
conv2_x	56×56	$\text{max pooling } 3 \times 3, \text{stride } 2$				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x*	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x*	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x*	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	<i>Average pooling</i> , 1000 camadas densas FC com função de ativação <i>Softmax</i>				

*Nota: Nas camadas conv3_1, conv4_1 e conv5_1 são utilizados *max pooling* 3×3 com *stride* 2.

Fonte: Adaptado de: (He *et al.*, 2016)

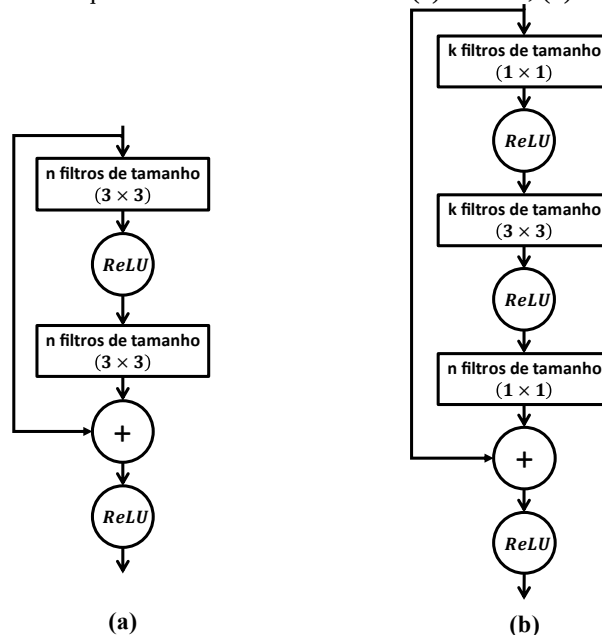
Para redes menos profundas, como a ResNet-18 e ResNet-34, os autores consideram o bloco residual ilustrado na Figura 21 (a), composto de duas camadas convolucionais 3×3 . Como sinais são somados, esse projeto de bloco residual requer que a saída tenha a mesma forma que a entrada.

Na iminência de alterar o número de canais, faz-se necessária a introdução de uma camada convolucional 1×1 , com o objetivo de transformar a entrada na forma desejada para possibilitar a operação de adição. Consequentemente, para redes mais profundas, como a ResNet-50, ResNet-101 e ResNet-152, os autores utilizam uma variante de bloco residual,

composto por uma pilha de três camadas ao invés de duas, denominado de bloco residual de gargalo (*Bottleneck Residual Block*) - ilustrado na Figura 21 (b). As três camadas são convoluções de dimensões 1×1 , 3×3 e 1×1 , respectivamente.

As camadas 1×1 são responsáveis, respectivamente, por reduzir e depois aumentar (restaurar) dimensões, deixando a camada 3×3 com um “gargalo” de entrada menor. Deste modo, o uso de um “gargalo” reduz o número de parâmetros e multiplicações de matrizes. A ideia é tornar os blocos residuais mais estreitos e possibilitar o aumento da profundidade da rede com menos parâmetros.

Figura 21 - Arquiteturas de blocos residuais: (a) comum; (b) *Bottleneck*.



Fonte: Adaptado de: (He *et al.*, 2016).

Com esta ideia de projeto modificado dos blocos residuais, as redes de maior profundidade mostraram um bom desempenho sem nenhum ajuste excessivo nos dados de treinamento.

2.2.7 ResNext

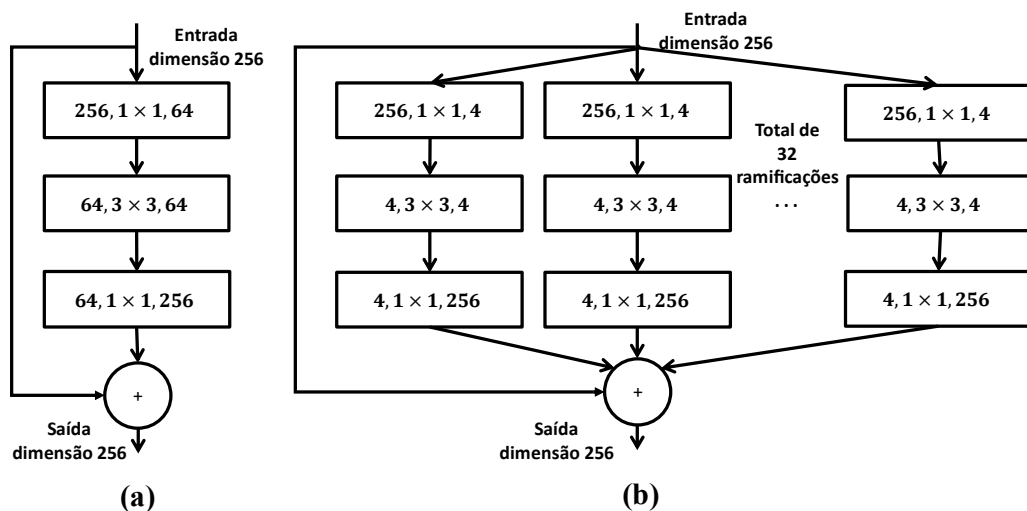
A arquitetura ResNext combina os pontos fortes dos projetos VGG, ResNet e *Inception*: composta pela repetição de blocos (como no VGG) que agrega um conjunto de transformações (como *Inception*) e as combina com as conexões residuais (da ResNet). Isso aplica um conjunto de transformações aos mapas de características de entrada e mescla com saídas resultantes antes de encaminhar as ativações de saída para o próximo módulo (Xie *et al.*, 2017).

A diferença entre o bloco ResNext ao módulo *Inception* é manifestada pelos três aspectos principais: Primeiro, contém um número maior de ramificações em comparação com o módulo *Inception*. Em segundo lugar, em contraste com diferentes tamanhos de filtros em diferentes ramificações da arquitetura *Inception*, a sequência de filtros em cada ramificação é idêntica a outras ramificações no módulo ResNext. Isso simplifica as escolhas de parâmetros da arquitetura ResNext, em comparação com o módulo *Inception*. Finalmente, o bloco ResNext contém conexões de *skip* que foram utilizadas nas redes residuais e consideradas cruciais no treinamento de redes muito profundas (Khan *et al.*, 2018).

A arquitetura geral do ResNext é equivalente ao design do ResNet mostrado na Tabela 1. A única diferença é que os blocos residuais são substituídos pelos blocos ResNext. Uma comparação entre os blocos é mostrada na Figura 22. Enquanto o bloco ResNet compreende uma única ramificação principal ao lado de uma conexão de *skip*, o bloco ResNext consiste em várias ramificações, formalmente denominadas de cardinalidade, e cada uma das quais implementa uma transformação. Juntamente com a profundidade e a largura, a cardinalidade é vista pelos autores como a próxima (*next*) dimensão vital nas redes neurais - originando o nome da rede (Xie *et al.*, 2017).

Para fins de comparação, o bloco ResNext proposto com 32 ramificações, Figura 22 (b), possui aproximadamente o mesmo número de parâmetros que o bloco ResNet ilustrado na Figura 22(a). Cada transformação é mostrada como: # número de canais de entrada, # dimensões do filtro, # número de canais de saída.

Figura 22 - Comparação entre exemplos de blocos residuais de dimensão 256: (a) ResNet; (b) ResNext.



Fonte: Adaptado de: (Xie *et al.*, 2017).

De modo geral, os dados são processados na sequência: divisão, transformação e união - como apresentado no módulo *Inception*. No exemplo da Figura 22, cada ramificação reduz

os dados para 4 canais, os transformam por convoluções 3×3 sem alterar o número de canais e, posteriormente, são devolvidos à dimensão 256. Esses tensores de 256 dimensões são somados e as conexões de *skip* adiciona a entrada ao resultado. Todo o processamento tem, aproximadamente, o mesmo número de parâmetros do que um bloco ResNet equivalente.

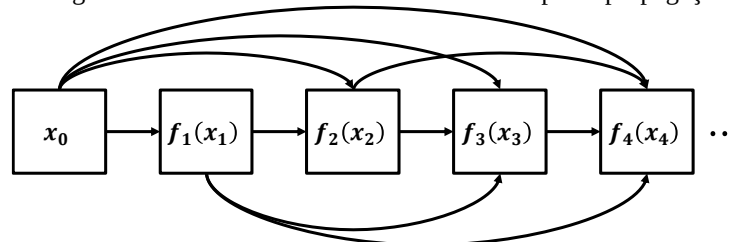
ResNeXt é um exemplo de como o projeto de redes neurais convolucionais evoluiu ao longo do tempo: existem alguns “truques” que levam a redes mais rápidas e precisas a um custo mais eficiente, *i.e.*, computação mais moderada ao parametrizar o tamanho das ativações (número de canais) (Zhang *et al.*, 2021).

2.2.8 DenseNet

A ResNet mudou significativamente a visão de como parametrizar as funções em redes profundas dado que o uso de conexões *skip*, sobretudo, evita o problema de fuga/explosão de gradiente e permitem o treinamento de redes muito profundas. A Rede Convolutiva Densa (DenseNet) estende essa ideia ao propagar a saída de cada camada para todas as camadas subsequentes, facilitando efetivamente a propagação da informação nas direções para frente e para trás durante o treinamento da rede.

O nome DenseNet surge do fato de que a dependência entre variáveis se torna bastante densa. A última camada dessa cadeia está densamente conectada a todas as camadas anteriores. As conexões são realizadas a partir de blocos densos e são representadas na Figura 23.

Figura 23 - Conexões entre blocos densos: etapa de propagação



Fonte: Adaptado de: (Zhang *et al.*, 2021).

Esta estratégia permite conexões entre todas as camadas da rede que possibilitem que elas “conversem” entre si e descubram automaticamente a melhor maneira de combinar recursos de vários estágios em uma rede profunda. Como uma das consequências destas conexões, as características das camadas iniciais são passadas diretamente para as camadas posteriores sem qualquer perda de informação. Em contrapartida, enquanto as redes convolucionais tradicionais com L camadas têm L conexões diretas, a DenseNet possuirá $\frac{L(L+1)}{2}$ conexões diretas. Um efeito - possivelmente contraintuitivo - deste padrão de

conectividade denso é que ele requer menos parâmetros do que as redes convolucionais tradicionais, uma vez que não há necessidade de reaprender mapas de características redundantes.

Além de ser caracterizada pelo padrão de conectividade entre suas camadas, a DenseNet também utiliza a operação de concatenação - ao invés do operador de adição no ResNet - para preservar e reutilizar recursos de camadas anteriores. Como resultado, é realizado um mapeamento de x para seus valores após aplicar uma sequência de funções cada vez mais complexa (Zhang *et al.*, 2021):

$$x \rightarrow [x, f_1(x), f_2([x, f_1(x)]), f_3([x, f_1(x), f_2([x, f_1(x)])]), \dots] \quad (5)$$

Devido à concatenação dos mapas de recursos de todas as camadas anteriores, o número de canais de entrada nas camadas subsequentes de um DenseNet aumenta rapidamente. Em aplicações convencionais, e para manter o processamento da rede viável, o número de canais de saída - também denominado taxa de crescimento (k) - para cada camada convolucional é relativamente pequeno; por exemplo, 6 ou 12.

A concatenação de mapas de características só pode ser realizada quando seus tamanhos espaciais coincidem. Portanto, uma CNN densamente conectada consiste em vários blocos, cada um dos quais possui conexões densas dentro das camadas e as operações de convolução com *pooling* e *stride* são executadas entre esses blocos para reduzir a entrada. As camadas que reduzem o tamanho das representações de recursos entre cada par de blocos DenseNet são chamadas de camadas de transição. As camadas de transição são implementadas como uma combinação de *batch normalization*, uma camada convolucional 1×1 e uma camada de *pooling* médio de janela 2×2 .

Em (Huang *et al.*, 2017) foram apresentadas variantes das arquiteturas DenseNets, na Tabela 2 são ilustradas as arquiteturas de DenseNet desenvolvidas para *ImageNet*. A taxa de crescimento para as primeiras três redes é $k = 32$ e a DenseNet-161 utiliza uma taxa de crescimento $k = 48$. Cada camada convolucional na tabela corresponde à sequência *batch normalization* seguido por ReLU seguido por Convolução. O número de blocos permanece o mesmo a cada variante, ou seja, 4, entretanto, a taxa de crescimento e o número de camadas de convolução são aumentados para projetar arquiteturas maiores.

Tabela 2 - Arquiteturas de DenseNets.

Camadas	Dimensão da saída	DenseNet-121 ($k = 32$)	DenseNet-169 ($k = 32$)	DenseNet-201 ($k = 32$)	DenseNet-161 ($k = 48$)
Convolução	112×112	Convolução 7×7 , <i>stride</i> 2			
Pooling	56×56	Max pooling 3×3 , <i>stride</i> 2			
Bloco Denso (1)	56×56	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 6$
Transição (1)	56×56	Convolução 1×1			
	28×28	Pooling médio 2×2 , <i>stride</i> 2			
Bloco Denso (2)	28×28	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 12$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 12$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 12$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 12$
Transição (2)	28×28	Convolução 1×1			
	14×14	Pooling médio 2×2 , <i>stride</i> 2			
Bloco Denso (3)	14×14	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 24$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 32$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 48$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 36$
Transição (3)	14×14	Convolução 1×1			
	7×7	Pooling médio 2×2 , <i>stride</i> 2			
Bloco Denso (4)	7×7	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 16$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 32$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 32$	$\begin{bmatrix} \text{conv. } 1 \times 1 \\ \text{conv. } 3 \times 3 \end{bmatrix} \times 24$
Classificação	1×1	Pooling médio global 7×7			
		Totalmente conectada de dimensão 1000 e função de ativação <i>Softmax</i>			

Fonte: (Huang *et al.*, 2017).

3 TRANSFERÊNCIA DE APRENDIZADO

Como consequência do sucesso e popularidade experimentada nos últimos tempos, houve muita melhoria no desempenho de sistemas com implementações por aprendizado de máquina. Todavia, também foram expostos outros desafios comumente encontrados nos paradigmas de aprendizagem.

Em geral, problemas complexos exigem muitos dados, mas a situação se complica quando precisa ser rotulado para aprendizado supervisionado. É muito difícil rotular uma grande quantidade de dados que pode exigir tempo, ferramentas e intervenção humana para rotulá-los, e mesmo depois de pronto, não há garantia de que todos os rótulos estejam corretos. Em muitos casos, para validar a qualidade dos rótulos em um conjunto de dados é necessária uma verificação dupla ou escolha do melhor de três rótulos (Ribani; Marengoni, 2019).

Adicionalmente, devido aos avanços das redes sociais, há muito conteúdo sendo compartilhado pelos usuários pela internet e estes dados não podem ser utilizados na elaboração de um banco de dados sem uma autorização explícita. Portanto, o respeito ao direito de uso de dados é outro dilema que deve ser considerado por empresas e pesquisadores que trabalham com inteligência artificial.

Estes desafios de processamento de dados e aprendizado de máquina resultaram na motivação da transferência de conhecimento de um modelo para outro, *i.e.*, a partir de um domínio de origem, treinar um modelo de alto desempenho para um domínio de destino relacionado.

O aprendizado por transferência tornou-se um subcampo proeminente no aprendizado de máquina que pode impulsionar o sucesso do aprendizado de máquina na pesquisa e na indústria. O princípio de transferência de conhecimento é fundamentado na ideia de que dois problemas devem ter bases de dados semelhantes, o que permitirá que um novo modelo emule, de outra forma análoga, a capacidade de mapear as principais características que relacionam a entrada à saída.

No aprendizado por transferência, em um primeiro momento se treina um modelo de origem em um conjunto de dados e uma tarefa de origem e, em seguida, os recursos aprendidos são transferidos para um segundo modelo a ser treinado em outro conjunto de dados e outra tarefa de destino. Esse processo tende a funcionar se os recursos forem gerais, ou seja, adequados para tarefas de base e de destino, em vez de específicos para a tarefa de destino (Yosinski *et al.*, 2014).

Com o uso de modelos pré-treinados para uma tarefa de origem, o aprendizado por transferência acelera o treinamento de novos modelos de aprendizado de máquina para uma tarefa de destino, uma vez que, o treinamento em milhões de imagens, por exemplo, pode requerer muito tempo além de *hardware* de alto custo.

Ademais, do ponto de vista biológico, o aprendizado por transferência é, na verdade, muito semelhante ao método de aprendizado humano - o conhecimento adquirido ao concluir uma tarefa também pode ser usado para resolver outras tarefas relacionadas. Quanto mais relacionadas forem as tarefas, mais eficaz será processo de transferência de conhecimento. Por exemplo, aprender andar de moto - possivelmente - será uma tarefa mais eficiente quando andar de bicicleta é um aprendizado já adquirido.

3.1 APRENDIZADO TRADICIONAL E APRENDIZADO TRANSFERIDO

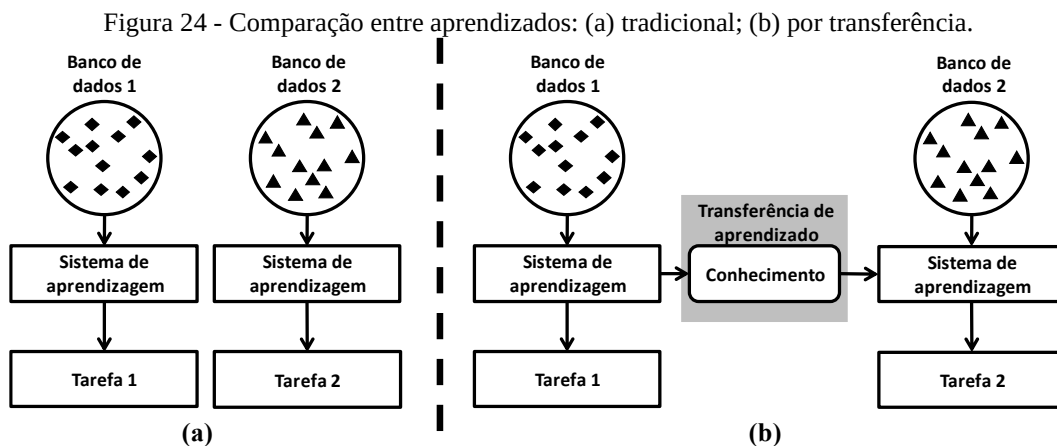
O aprendizado de máquina tradicional é uma abordagem que atua para que modelos aprendam conjuntos de dados específicos. Nesses casos, o modelo deve ser construído e treinado do zero e requer muitos dados. Normalmente, nesta abordagem não ocorre retenção de conhecimento para uso posterior, ou seja, enquanto o aprendizado para alguma tarefa é realizado, nenhum conhecimento previamente aprendido é considerado.

O aprendizado por transferência, por outro lado, permite transferir o conhecimento existente de um treinamento anterior - de origem - para atuar como um conhecimento inicial de entrada em uma nova configuração de tarefa de destino - idealmente, com semelhança mínima entre eles. Deste modo, um aprendizado de uma nova tarefa também depende dos aprendizados de tarefas aprendidas anteriormente.

A Figura 24 ilustra a comparação entre um processo de aprendizagem tradicional de duas tarefas diferentes e as mesmas tarefas aprendidas usando um método de aprendizagem por transferência, em que a tarefa 1 é totalmente treinada usando um conjunto de dados e a tarefa 2 utiliza o conhecimento obtido na tarefa 1 para melhorar a precisão e aprender mais rápido.

A abordagem por transferência de conhecimento torna o processo de treinamento mais rápido, com maior grau de precisão e evita a necessidade de um conjunto de dados de treinamento de grande porte, *i.e.*, aprendizado por transferência de conhecimento já funciona bem com dados de treinamento menores, com resultados acelerados (Gupta; Bhardwaj; Sharma, 2020). Yosinski *et al.* (2014) demonstraram que a inicialização com recursos transferidos pode melhorar o desempenho da generalização do sistema, mesmo após um ajuste

fino substancial em uma nova tarefa, o que pode ser uma técnica geralmente útil na melhoria geral do desempenho.



Fonte: Elaboração do próprio autor.

3.2 DEFINIÇÕES E NOTAÇÕES

O trabalho de Pan; Yang (2010) se tornou uma referência, pois, introduziu, de forma notável, definições e notações na área de transferência de conhecimento; também utilizada por outros autores, *e.g.*, os trabalhos de Ribani; Marengoni (2019) e Weiss; Khoshgoftaar; Wang (2016), também citados neste capítulo.

Para manter a consistência com outras pesquisas, estas mesmas definições e notações serão utilizadas. Estas definições - e suas notações - fundamentam a transferência de aprendizado a partir de dois conceitos: o domínio e a tarefa.

Um domínio $\mathcal{D}$, que está relacionado à distribuição dos dados a serem usados para treinamento, é composto por duas partes: um espaço de características $\mathcal{X}$ e uma distribuição de probabilidade marginal $P(X)$, onde $X = \{x_1, \dots, x_n\} \in \mathcal{X}$. Neste caso, $\mathcal{X}$ é todo o espaço de características possíveis, x_i é a i -ésima característica contida em algumas amostras dos dados de treinamento e X é uma amostra particular de aprendizado com um subconjunto de características de $\mathcal{X}$. Para considerar dois domínios como diferentes, eles devem ter diferentes espaços de características ou diferentes distribuições de probabilidade marginal.

Uma tarefa $\mathcal{T}$ é definida por duas partes: um conjunto de todos os rótulos possíveis $\mathcal{Y}$ e um modelo ou função preditiva $f(\cdot)$ que prevê um rótulo correspondente com base em um determinado domínio $\mathcal{D}$. A tarefa é denotada por $\mathcal{T} = \{\mathcal{X}, f(\cdot)\}$ e pode ser aprendida a partir dos dados de treinamento, que consiste em uma coleção de pares $\{x_i, y_i\}$, onde $x_i \in \mathcal{X}$ e $y_i \in \mathcal{Y}$. A função $f(\cdot)$ é usada para prever um rótulo y ou $f(x)$ de uma instância x .

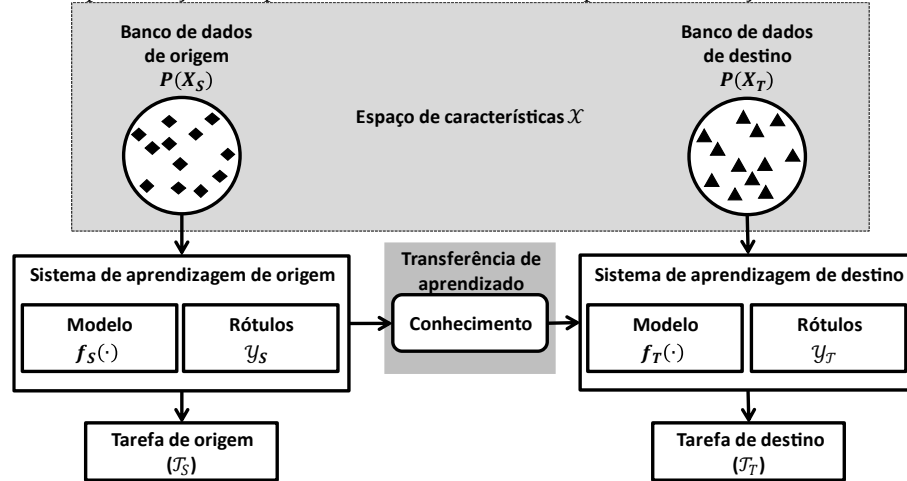
Transferência de aprendizado é definida formalmente como: dado um domínio de origem $\mathcal{D}_S$ e uma tarefa de aprendizagem de origem $\mathcal{T}_S$, um domínio de destino $\mathcal{D}_T$ e uma

tarefa de aprendizagem de $\mathcal{T}_T$, a transferência de aprendizado é definida como um processo que ajudará a melhorar a função $f_T(\cdot)$ no aprendizado da tarefa de destino com base no conhecimento obtido de $\mathcal{D}_S$ e $\mathcal{T}_S$, onde $\mathcal{D}_S \neq \mathcal{D}_T$ e/ou $\mathcal{T}_S \neq \mathcal{T}_T$.

Quando existe alguma relação, explícita ou implícita, entre os espaços de características dos dois domínios, os domínios de origem e destino estão relacionados. Por outro lado, quando os domínios de origem e destino são os mesmos ($\mathcal{D}_S = \mathcal{D}_T$) e as tarefas de origem e destino são as mesmas ($\mathcal{T}_S = \mathcal{T}_T$), isto se torna um problema de aprendizado de máquina tradicional.

Na Figura 25 é apresentado o processo de transferência de aprendizagem da Figura 24 com as definições de domínio e tarefa. Neste caso, os domínios de origem e de destino compartilham o mesmo espaço de recursos enquanto as distribuições de probabilidade marginal são diferentes.

Figura 25 - Representação do aprendizado de transferência a partir da definição de domínio e tarefa.



Fonte: Adaptado de: (Ribani; Marengoni, 2019).

3.3 CONFIGURAÇÕES DE TRANSFERÊNCIA DE APRENDIZADO

Baseado nas definições de transferência de aprendizagem e em diferentes cenários existentes entre origem e destino, (Pan; Yang, 2010) categorizam a transferência de conhecimento em três configurações: indutiva, transdutiva e não supervisionada.

3.3.1 Transferência de conhecimento indutiva

A transferência de conhecimento indutiva é utilizada em casos que as tarefas de origem e destino são diferentes enquanto os domínios podem estar, ou não, relacionados. A implementação da transferência de aprendizagem por indução trata de um cenário onde o domínio funcional deve ser compartilhado pelos dois modelos (origem e destino). Neste cenário, para induzir o modelo preditivo de destino $f_T(\cdot)$ são necessários alguns dados

rotulados na tarefa de destino. Deste modo, essa configuração tenta melhorar a tarefa de destino a partir da indução dos vieses do domínio de origem.

Um tipo comum de aprendizagem indutiva é a ideia de obter um modelo de aprendizado profundo pré-treinado usando um domínio e uma tarefa de origem e realizar um ajuste fino para diferentes camadas na tarefa de destino. Por exemplo, obter um modelo pré-treinado com o conjunto de dados ImageNet e ajustá-lo para reconhecer diferentes classes de objetos (Ribani; Marengoni, 2019).

Pan; Yang (2010) expandem que, dependendo da disponibilidade de dados rotulados da origem, a transferência de conhecimento indutiva pode ser dividida em dois casos: quando muitos dados rotulados no domínio de origem estão disponíveis ou quando nenhum dado rotulado no domínio de origem está disponível.

No primeiro caso, a configuração de aprendizado por transferência indutiva é semelhante à configuração de aprendizado multitarefa. No entanto, a configuração de aprendizagem por transferência indutiva visa apenas alcançar alto desempenho na tarefa de destino, transferindo conhecimento da tarefa de origem, enquanto o aprendizado multitarefa tenta aprender, simultaneamente, a tarefa de destino e a origem.

Já no segundo caso, a configuração de aprendizagem por transferência indutiva é semelhante à configuração de aprendizagem autodidata. Neste caso, as informações do domínio de origem não podem ser usadas diretamente, pois os rótulos podem ser diferentes, mas os dados de treinamento podem ser usados para melhorar o desempenho na tarefa de destino. A tarefa de origem pode ser treinada para reconhecer padrões nos dados não rotulados e, em seguida, esses padrões podem ser usados em uma tarefa de aprendizado supervisionada de destino.

Para uma compreensão mais simples, pode ser visto como um processo de duas etapas: primeiramente, a tarefa de origem é alcançada por meio de autoaprendizagem, marcando os rótulos de domínio de origem no processo. Isso estabelece um esquema de rotulagem de dados de recursos em relação às inferências correspondentes feitas em relação à tarefa de origem. Em seguida, este conhecimento e mapeamento obtido por meio das interações para a tarefa de origem são estendidos para as operações da tarefa de destino, melhorando seu desempenho.

3.3.2 Transferência de conhecimento transdutiva

Na aprendizagem por transferência transdutiva, as tarefas de origem e destino são as mesmas, enquanto os domínios de origem e destino são diferentes. Nessa situação, nenhum

dado rotulado no domínio de destino está disponível enquanto muitos dados rotulados no domínio de origem estão disponíveis.

Além disto, de acordo com diferentes situações entre os domínios de origem e de destino, pode-se categorizar a configuração de aprendizagem por transferência transdutiva em dois casos: quando os espaços de características entre os domínios são diferentes, $\chi_S \neq \chi_T$, ou quando os espaços de características entre os domínios são os mesmos, $\chi_S = \chi_T$, mas as distribuições de probabilidade marginal dos dados de entrada são diferentes, $P(X_S) \neq P(X_T)$.

Um exemplo de transferência transdutiva é o uso de imagens sintéticas em uma tarefa de origem para aumentar o desempenho em uma tarefa de destino com imagens do mundo real.

3.3.3 *Transferência de conhecimento não supervisionada*

Na aprendizagem por transferência não supervisionada, a tarefa de destino é diferente da tarefa de origem, porém relacionada. No entanto, o aprendizado de transferência não supervisionada foca na resolução de tarefas de aprendizado não supervisionado no domínio alvo, como agrupamento, redução de dimensionalidade e estimativa de densidade. Neste caso, não há dados rotulados disponíveis nos domínios de origem e destino no treinamento.

A relação entre as diferentes configurações de aprendizagem de transferência e as áreas relacionadas está resumida na Tabela 3.

Tabela 3 - Relação entre as diferentes configurações de aprendizagem de transferência.

Configurações de aprendizagem de transferência	Rótulos no domínio de origem	Rótulos no domínio de destino	Áreas relacionadas	Tarefas
Transferência de conhecimento indutiva	Não disponível	Disponível	Aprendizado multitarefa	Regressão e classificação
	Disponível	Disponível	Aprendizagem autodata	Regressão e classificação
Transferência de conhecimento transdutiva	Disponível	Não disponível	Adaptação de domínio, viés de seleção de amostra, mudança de covariável	Regressão e classificação
Transferência de conhecimento não supervisionada	Não disponível	Não disponível		Agrupamento e redução de dimensionalidade

Fonte: Adaptado de: (Pan; Yang, 2010).

3.4 ABORDAGENS DE APRENDIZAGEM DE TRANSFERÊNCIA

(Pan; Yang, 2010) introduzem três configurações apresentadas na Seção 3.3 em quatro abordagens que podem ser aplicadas em termos de “O que transferir”.

A primeira abordagem é chamada de transferência de instância, um caso de uso muito comum em que partes do domínio de origem podem ser reutilizadas no aprendizado do domínio de destino para melhorar a tarefa de destino. Este processo pode ser chamado de “reponderação” - do inglês *re-weighting*.

A segunda abordagem é chamada de transferência de representação de características. Nesta abordagem utilizam-se as representações de características aprendidas na tarefa de origem para melhorar a tarefa de destino. O conhecimento é codificado nesta representação de características. Uma aplicação comum desta abordagem é o uso de modelos pré-treinados com o conjunto de dados ImageNet para o treinamento de outros modelos para novas tarefas.

Um terceiro caso é referido como abordagem de transferência de parâmetros, que assume que as tarefas de origem e as tarefas de destino compartilham alguns parâmetros ou distribuições anteriores de hiperparâmetros entre os modelos. O conhecimento transferido é, na verdade, os parâmetros compartilhados entre os modelos.

A última abordagem é a transferência de conhecimento relacional que, diferente das três acima, trata da transferência de conhecimento entre domínios relacionais. A suposição básica por trás deste contexto é que haja alguma semelhança entre os dados nos domínios de origem e destino. Deste modo, o conhecimento a ser transferido é a relação entre os dados. Pan e Yang (2010) citam que técnicas de aprendizado relacional estatístico dominam esse contexto e Ribani; Marengoni (2019) complementam que é comum que modelos que usam dados de redes sociais implementem aprendizado de transferência relacional.

Na Tabela 4 estão representadas as relações entre as diferentes configurações de aprendizagem de transferência e o que transferir.

Tabela 4 - Diferentes abordagens usadas em diferentes configurações.

	Transferência de conhecimento indutiva	Transferência de conhecimento transdutiva	Transferência de conhecimento não supervisionada
Transferência de instância	Se aplica	Se aplica	Não se aplica
Transferência de representação de características	Se aplica	Se aplica	Se aplica
Transferência de parâmetros	Se aplica	Não se aplica	Não se aplica
Transferência de conhecimento relacional	Se aplica	Não se aplica	Não se aplica

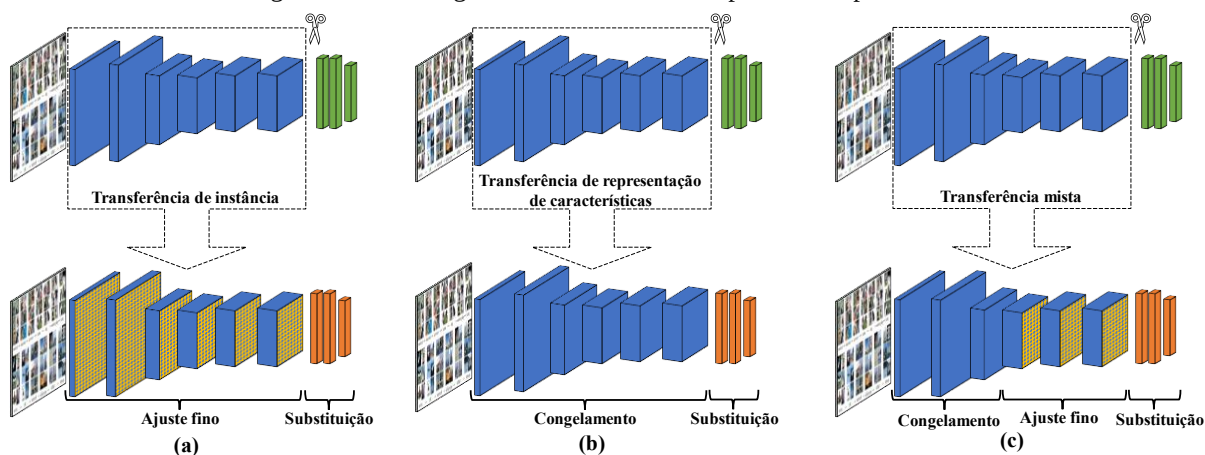
Fonte: Adaptado de: (Pan; Yang, 2010).

3.5 TRANSFERÊNCIA DE APRENDIZADO PARA REDES CONVOLUCIONAIS

Normalmente, as arquiteturas de redes convolucionais são inicialmente construídas a partir de uma sequência de camadas de convolução e *pooling*, para aprender representações hierárquicas dos dados e, portanto, estas camadas funcionam como um extrator de recursos. Conforme ilustrado na Figura 4, as redes convolucionais exibem um fenômeno comum: as primeiras camadas tendem a aprender representações básicas, que se assemelham a filtros Gabor (Mehrotra; Namuduri; Ranganathan, 1992), como bordas e cantos, enquanto as camadas posteriores aprendem representações mais complexas, como partes de objetos. Por outro lado, as camadas finais, que normalmente contêm apenas neurônios totalmente conectados, dependem muito do conjunto de dados e são responsáveis por realizar tarefas específicas, como classificação ou regressão (Yosinski *et al.*, 2014).

Este padrão de arquitetura permite o uso de um modelo pré-treinado para realizar a transferência de aprendizado. Por exemplo, é possível substituir as últimas camadas e realizar um ajuste fino dos parâmetros do modelo para uma tarefa de destino específica. Neste caso de implementação, trata-se de uma abordagem de transferência de instância. Em outra perspectiva, é possível utilizar o modelo pré-treinado e congelar as primeiras camadas, deste modo, o modelo de origem funcionará como um extrator de características para as últimas camadas na tarefa de destino. Neste caso, trata-se de uma implementação da abordagem de representação de características. Além disso, também é possível usar uma mistura de ambas as abordagens, selecionando quais camadas congelar ou ajustar, o que pode ser útil dependendo dos domínios de origem e destino.

Figura 26 - Abordagens de transferência de aprendizado para CNNs.



Fonte: Elaboração do próprio autor.

A escolha de ajustar ou não as primeiras camadas da rede de destino dependem da distribuição dos dados, do tamanho do conjunto de dados de destino e do número de

parâmetros nas camadas. Ribani; Marengoni (2019) apontam que, se o domínio de destino for semelhante ao de origem, o congelamento de camadas dará bons resultados por causa das representações de recursos comuns contidas entre os domínios. No entanto, se o domínio de destino contém recursos diferentes, provavelmente será necessário descongelar e ajustar algumas ou todas as camadas para que o modelo se adapte à nova distribuição contida no domínio de destino.

Adicionalmente, o tamanho do conjunto de dados e o número de parâmetros devem ser considerados de modo a evitar o *overfitting*, que tende a ocorrer com grandes modelos de rede e/ou conjuntos de dados pequenos. Desta forma, se o conjunto de dados de destino for pequeno e o número de parâmetros for grande, deve-se ter cautela ao realizar o ajuste fino das camadas da rede. Por outro lado, se o conjunto de dados de destino for grande ou o número de parâmetros for pequeno, de modo que o *overfitting* não seja um problema, o ajuste fino pode ser realizado para melhorar o desempenho em uma nova tarefa.

3.5.1 Estratégias de transferência de aprendizado

Huang; Le (2021) sintetizam quatro estratégias de transferência de aprendizado de redes convolucionais que apontam quando e quais camadas devem ser congeladas ou ajustadas. Estas estratégias são baseadas em duas características dos dados de treinamento: o tamanho e a semelhança entre os domínios.

A primeira estratégia aborda casos em que se deseja treinar modelos de rede com conjuntos de dados pequenos e amostras de treinamento semelhantes. Como o novo conjunto de dados possui semelhança com o conjunto de dados usado para o modelo pré-treinado, eles possuem recursos semelhantes nas camadas ocultas da rede, principalmente nas camadas mais profundas. Portanto, as camadas para extração de recursos não precisam ser alteradas e, para aprender uma nova tarefa, é necessário apenas fazer alterações na última camada que trata dos recursos de saída do modelo de rede. O novo modelo de rede é construído com base na arquitetura do modelo pré-treinado, removendo a última camada e adicionando novas camadas para a nova tarefa.

A segunda estratégia trata de casos de treinamento de redes com conjuntos de dados pequenos e diferentes amostras de treinamento. Neste caso, para evitar o *overfitting*, os pesos do modelo pré-treinado devem ser mantidos inalterados. Como as amostras do novo conjunto de dados são diferentes daquelas do conjunto de dados usado para o modelo pré-treinado, apenas nas camadas mais rasas possuem características semelhantes, enquanto suas características nas camadas mais profundas do modelo da rede são muito diferentes. Neste

caso, para construir um modelo de rede para aprender uma nova tarefa, baseado na arquitetura do modelo pré-treinado, é necessário manter apenas as camadas mais rasas, descartar as camadas mais profundas e substituir a camada de saída por uma nova.

A terceira estratégia aborda treinamento de redes com conjuntos de dados grandes e amostras de treinamento semelhantes. Neste caso, é possível treinar todo o modelo de rede pré-treinado. Como o novo conjunto de dados tem semelhança com o conjunto de dados usado para o modelo pré-treinado, os recursos são semelhantes nas camadas ocultas da rede - especialmente nas camadas mais profundas. Portanto, as camadas para extração de recursos não precisam ser alteradas e, para aprender uma nova tarefa, é necessário apenas fazer alterações na última camada que trata a saída do modelo de rede. Deste modo, o novo modelo será construído com base na arquitetura do modelo pré-treinado, substituindo a última camada por uma nova e os pesos das camadas recém-adicionadas, bem como algumas ou todas as outras camadas do novo modelo de rede, podem ser otimizados para obter melhor desempenho.

A quarta estratégia consiste em treinar modelos de rede com conjunto de dados grandes e amostras de treinamento diferentes. O uso de um conjunto de dados maior para treinar um modelo de rede pode evitar o problema de *overfitting*. Portanto, todo o modelo de rede pré-treinado ou um novo modelo de rede pode ser treinado no novo conjunto de dados para melhorar o desempenho. Neste caso é possível utilizar dois métodos de treinamento: a criação de um modelo de rede ou o uso de um modelo pré-treinado. Devido à diferença entre o novo conjunto de dados e o conjunto de dados utilizado para o treinamento do modelo pré-treinado, as características do modelo pré-treinado não serão semelhantes às características da nova rede na maioria das camadas e, portanto, pode ocorrer transferência negativa de aprendizado. Huang; Le (2021), no entanto, destacam que, na prática, muitas vezes ainda é benéfico inicializar com pesos de um modelo pré-treinado para treinar um novo modelo de rede.

3.6 TRANSFERÊNCIA NEGATIVA

A transferência negativa, no que diz respeito ao aprendizado por transferência, ocorre quando a informação aprendida de um domínio de origem tem um efeito prejudicial no modelo de destino. Formalmente, a transferência negativa ocorre quando o desempenho de $f_{T_1}(\cdot)$, treinado apenas com $\mathcal{D}_T$, é maior que o desempenho de $f_{T_2}(\cdot)$, treinado com um processo de transferência aprendizado com a combinação de $\mathcal{D}_T$ e $\mathcal{D}_S$ (Weiss; Khoshgoftaar; Wang, 2016).

Para Ribani; Marengoni (2019), no campo da aprendizagem por transferência, a transferência negativa é uma questão em aberto e há uma necessidade de estudar a viabilidade de transferência de conhecimento entre domínios ou tarefas de origem e domínios ou tarefas de destino. Pode haver várias razões para causa da transferência negativa, como: casos em que a tarefa de origem não está suficientemente relacionada à tarefa de destino, ou se o método de transferência não pode alavancar muito bem o relacionamento entre as tarefas de origem e destino.

4 TEORIA DA RESSONÂNCIA ADAPTATIVA

Introduzida por Stephen Grossberg em 1976, a Teoria da Ressonância Adaptativa (ou ART, de *Adaptive Resonance Theory*) foi desenvolvida com o propósito de solucionar o dilema da plasticidade e estabilidade. Esta teoria é inspirada no processamento da informação cognitiva humana em como um cérebro direciona a atenção para reconhecer e aprender padrões em ambientes de constante mudança à medida que novas informações são apresentadas (Grossberg, 1976a, 1976b). Por consequência, os modelos baseados nesta teoria são capazes de distinguir entre eventos familiares e não familiares, bem como entre eventos esperados e inesperados (Carpenter; Grossberg, 1987a).

O termo “adaptativo” faz referência às características de plasticidade da rede, *i.e.*, aprender e refinar o conhecimento ao ser exposta a novos padrões de entrada. Por sua vez, a “ressonância” está correlacionada à sua estabilidade, ou seja, não perder informações já aprendidas.

A teoria afirma que a ressonância regula o aprendizado e, em virtude de seu mecanismo de vigilância, que administra a inclusão de novas entradas em cada grupo, sendo que quando certa entrada não se assemelha a nenhum dos grupos já existentes gera-se um novo grupo relativo à respectiva. O problema de esquecimento catastrófico (tratado na Seção 1.1) não é presente em modelos de redes baseados em ART.

O aprendizado das redes da família ART é por treinamento online, ou seja, os pesos sinápticos são atualizados a cada apresentação de um novo padrão de entrada. Entretanto, como supracitado, essas atualizações dos pesos não interferem no conhecimento anteriormente adquirido, portanto, apresentam melhor estabilidade do que outras técnicas de aprendizado, como o popular algoritmo de retropropagação (Carpenter; Grossberg, 1987a).

Grossberg e seus colaboradores (em destaque para a pesquisadora Gail Carpenter) desenvolveram muitas arquiteturas de redes ARTs. As primeiras redes ART usavam aprendizagem não supervisionada ou auto-organizada. Entre elas se destacam: a ART1 (Carpenter; Grossberg, 1987a), que possui a capacidade de reconhecer padrões de entrada binários de forma arbitrária; a ART2 (Carpenter; Grossberg, 1987b) desenvolvida para detectar regularidades nas sequências randômicas de entradas tanto binárias quanto analógicas. Não sendo muito popular, pois possui uma arquitetura computacionalmente cara e apresenta dificuldades na seleção de parâmetros, mas que inspirou outras redes para superar seus problemas; como a ART *Fuzzy* (Carpenter; Grossberg; Rosen, 1991a) que engloba, em sua arquitetura, cálculos baseados na lógica *Fuzzy* (também conhecida como lógica nebulosa).

Nos anos posteriores, modelos que usavam aprendizagem supervisionada foram propostos, ainda que também contemplassem a auto-organização durante o treinamento. Destaca-se: a ARTMAP (Carpenter; Grossberg; Reynolds, 1991) - fundamental para todas as outras redes ART com treinamento supervisionado - esta arquitetura é composta pelo campo inter-ART que interconecta dois módulos ART e é capaz de identificar padrões de entradas binários ou analógicos; e a ARTMAP *Fuzzy* (Carpenter *et al.*, 1992) que é, basicamente, uma rede ARTMAP que utiliza lógica *Fuzzy* para associar os padrões de entrada e saída.

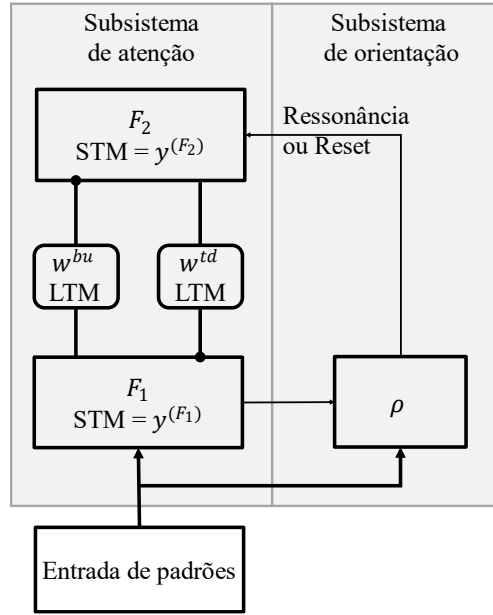
Apesar de, em sua essência, os modelos elementares baseados na ART serem predominantemente usados para aplicações de aprendizado não supervisionado, eles também estabelecem as bases para construir sistemas complexos, capazes de executar as modalidades de aprendizado supervisionado e aprendizado por reforço, bem como alguns modelos que cruzam esses limites e/ou combinam várias modalidades de aprendizado (da Silva; Elnabarawy; Wunsch, 2019).

Além dos modelos elementares, existem diversos outros modelos de redes da família ART que têm sido usados em várias aplicações no mundo real. O artigo de da Silva; Elnabarawy; Wunsch (2019) apresenta um resumo dos principais modelos de redes neurais baseados na ART utilizados com grande escala em aplicações de engenharia, destaca a atividade em pesquisas na área e fornece uma visão do potencial da Teoria da Ressonância Adaptativa.

4.1 CARACTERÍSTICAS E DINÂMICA DAS REDES ART

A arquitetura elementar de um modelo de rede neural ART - ilustrada na Figura 27 - é composta de dois subsistemas: o de atenção e de orientação. O subsistema atencional possui duas camadas de neurônios totalmente conectadas: os campos F_1 e F_2 . O primeiro campo tem função de comparação; já o segundo é onde ocorre o reconhecimento. Alguns modelos ART apresentam procedimentos de pré-processamento das amostras de entrada por outra camada, anterior a F_1 , denominada de campo de entrada F_0 .

Figura 27 - Anatomia da rede ART elementar.



Fonte: Adaptado de: (Carpenter; Grossberg, 1987a).

As atividades desenvolvidas nas duas camadas, F_1 e F_2 , são chamadas de memórias de curto prazo-*short-term memory* (STM) e são conectadas de forma bidirecional por pesos adaptáveis, denominados de memória de longo prazo - *long-term memory units* (LTMs).

No modo direcional *feedforward*, a saída STM da camada F_1 ($y^{(F_1)}$) propaga as amostras de entrada $x \in R^d$ para a camada F_2 através das LTMs de baixo para cima (*bottom-up*) w^{bu} .

Os modelos ART são redes competitivas, auto-organizadas, dinâmicas e modulares, portanto, quando uma amostra x é apresentada, uma competição do tipo *winner take all* (WTA) ocorre sobre as categorias na camada F_2 . Em seguida, é escolhido o neurônio J que otimiza a função de ativação daquele modelo entre os nós. Por exemplo, o neurônio que maximiza alguma medida de similaridade T para a amostra apresentada, *i.e.*:

$$J = \arg \max_j (T_j) \quad (6)$$

No modo direcional *feedback*, a camada F_1 compara a expectativa aprendida descendente de F_2 , ponderada por uma LTM de cima para baixo (*top-down*) w^{td} , com a entrada x ascendente (*bottom-up*) e o resultado $y^{(F_1)}$ é enviado para o subsistema orientador. Destaca-se que nem todos os modelos elementares de ART possuem as LTMs com as partes ascendentes e descendentes independentes; no entanto, caso as duas partes forem presentes, a LTM associada a uma categoria j é definida como: $w_j = \{w_j^{bu}, w_j^{td}\}$, $j = 1, \dots, N$. Tal que N representa o número de categorias em F_2 .

Uma categoria vencedora representa uma hipótese que é testada pelo subsistema de orientação para determinar a adequação da categoria selecionada. Se a confiança em tal hipótese for maior que o limite mínimo (denominado de parâmetro de vigilância (ρ), onde $0 < \rho \leq 1$), a rede neural entra em um estado de ressonância e o aprendizado é permitido, ou seja, é realizada a adaptação das LTMs. Caso contrário, a categoria J é inibida, a pesquisa é retomada e a próxima categoria de classificação mais alta é selecionada. Se nenhuma categoria satisfizer a condição de ressonância necessária, então uma nova é criada para codificar a amostra de entrada apresentada. Categorias inativas sempre serão criadas de modo a atender o critério de vigilância.

O parâmetro de vigilância estipula o nível de seletividade do sistema: um sistema com parâmetro de vigilância unitário anula a capacidade de generalização de rede visto que, a ressonância só ocorrerá quando o vetor de entrada possuir características exatas às de uma categoria já conhecida. Por outro lado, um treinamento com ρ pequeno, produzirá uma rede com capacidade de generalização maior, porém, incapaz de assimilar características individuais de cada classe e, conseqüentemente, comprometer a precisão na classificação.

Resumindo, é a partir dos padrões de entrada que o subsistema de atenção indica a sua categoria equivalente, todavia, o subsistema atencional isolado é incapaz de manter, simultaneamente, representações estáveis de categorias familiares e criar categorias para padrões desconhecidos. O subsistema orientador é quem regula tanto os mecanismos de busca quanto os de aprendizagem, inibindo ou permitindo que as categorias ressoem. Os dois subsistemas são funcionalmente complementares e interagem entre si de modo a processar padrões familiares e não familiares pela dinâmica: ativação, correspondência, ressonância e aprendizado.

4.2 ART 1

O primeiro modelo, a ART 1, introduzido em 1987 por Gail Carpenter e Stephen Grossberg, foi a implementação da teoria defendida por Grossberg usada para aplicações de engenharia e se tornou uma inspiração para criação de modelos subsequentes. Como características principais, possui treinamento não supervisionado e a capacidade de reconhecimento de padrões de entrada binários de forma aleatória (Carpenter; Grossberg, 1987a).

As camadas F_1 e F_2 da ART 1 fazem conexões bidirecionais e, portanto, as categorias são parametrizadas com vetores de pesos adaptativos (LTMs) tanto de baixo para cima quanto

de cima para baixo e, portanto, a LTM associada a uma categoria j é definida como $w_j = \{w_j^{bu}, w_j^{td}\}$, $j = 1, \dots, N$ (Carpenter; Grossberg, 1987a).

Quando uma amostra binária x de dimensionalidade d é apresentada à ART 1, um sinal de um neurônio j em F_1 é transportado ao longo de um caminho para F_2 com a ponderação pela LTM *bottom up* (w_j^{bu}), conseqüentemente, a função de ativação de cada categoria é computada como:

$$T_j = \|x \cap w_j^{bu}\|_1 = \sum_{i=1}^d x_i \cdot w_{ji}^{bu} \quad (7)$$

Onde $\cap$ representa a lógica binária AND e $\|x \cap w_j^{bu}\|_1$ é a normalização L_1 .

Na camada F_2 , o nó selecionado pela competição WTA será a categoria que melhor representa o vetor de padrão de entrada e os demais nós permanecerão inativos. Em um caso especial deste processo, será aquele neurônio que recebe a maior entrada T_j , neste caso, a saída da atividade na camada F_2 (STM) será:

$$y_j^{(F_2)} = \begin{cases} 1, & \text{se } T_j = \max_j \{T_j\} \\ 0, & \text{caso contrário} \end{cases} \quad (8)$$

Depois que a saída da atividade F_2 surge, um sinal é enviado de F_2 para F_1 e ponderado por pesos w_j^{td} , por consequência, um novo padrão de atividade STM, denominado de $y^{(F_1)}$, é gerado em F_1 :

$$y^{(F_1)} = \begin{cases} x, & \text{se } F_2 \text{ está inativa} \\ x \cap w_j^{td}, & \text{caso contrário} \end{cases} \quad (9)$$

O nó com maior padrão de atividade J é comparado com o parâmetro de vigilância predeterminado ($\rho \in [0,1]$) pelo teste de ressonância e duas situações podem ocorrer:

Se $\frac{\|x \cap w_J^{td}\|_1}{\|x\|_1} \geq \rho$, a categoria ativa J é aceita e os pesos sinápticos serão atualizados de modo a incorporar o novo conhecimento.

Caso contrário, a categoria ativa J não será aceita e será resetada de modo a atualizar $T_j = 0$ para que outro nó seja selecionado na camada F_2 .

A etapa de aprendizado iniciará fundamentada no caso 0 supracitado e os pesos serão atualizados pelas equações (10) e (11):

$$w_j^{td}(new) = I \cap w_j^{td}(old) \quad (10)$$

$$w_j^{bu}(new) = \frac{L}{L - 1 + \|w_j^{td}(new)\|_1} w_j^{td}(new) \quad (11)$$

De modo que $L \geq 1$ é um parâmetro definido pelo usuário, utilizado para ajustar a velocidade de treinamento (Serrano-Gotarredona; Linares-Barranco, 1996).

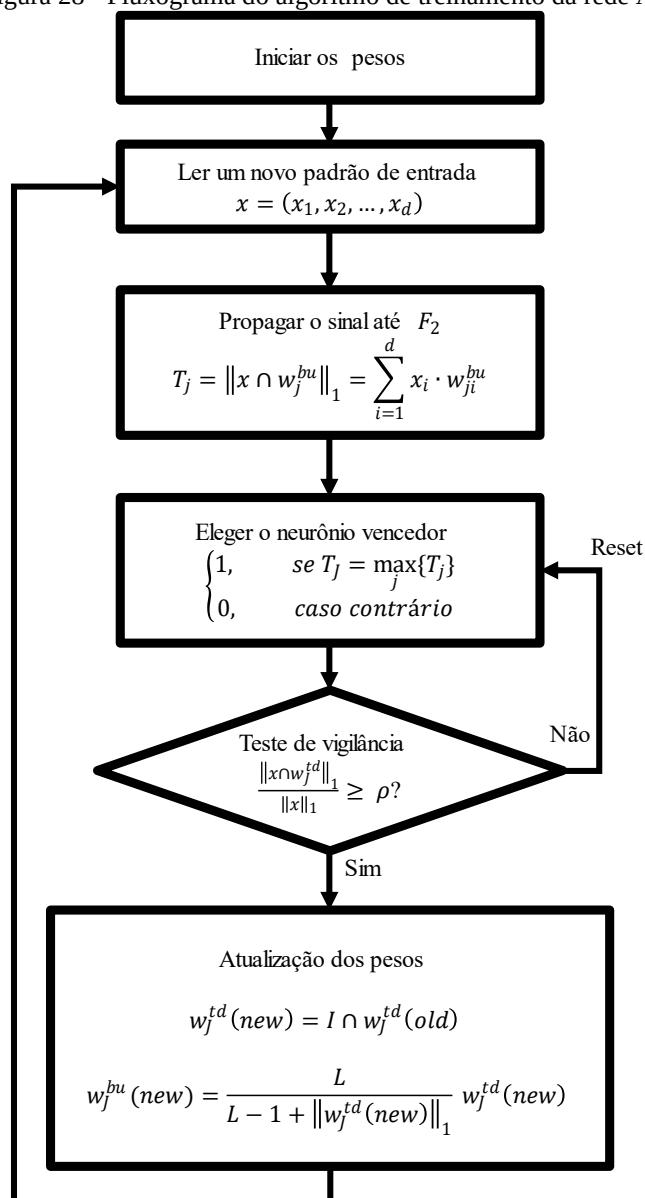
Se a entrada x for muito diferente de qualquer uma das informações aprendidas anteriormente, um nó não ativo em F_2 será selecionado e o aprendizado de uma nova categoria será iniciado tal que:

$$w^{td} = \vec{1} \quad (12)$$

$$w^{bu} = \frac{L}{L - 1 + d} w^{td} \quad (13)$$

O fluxograma de implementação do algoritmo de treinamento da rede ART1 é representado na Figura 28.

Figura 28 - Fluxograma do algoritmo de treinamento da rede ART 1.



Fonte: Adaptado de: Carpenter; Grossberg (1987a).

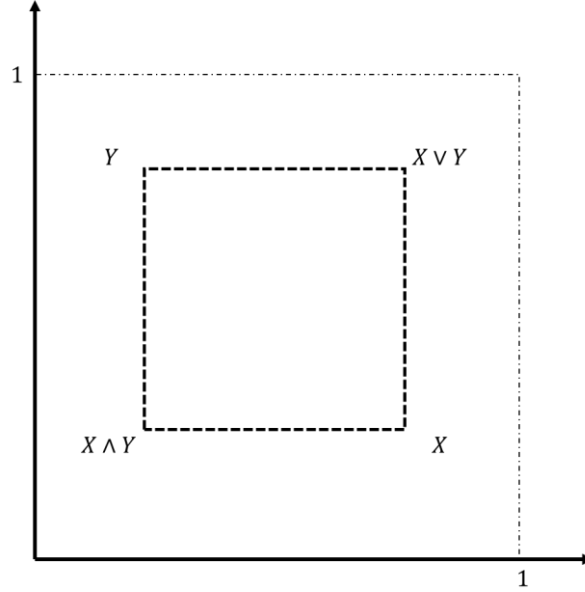
4.3 ART FUZZY

A rede ART *Fuzzy* é provinda da conexão do sistema ART com a lógica *Fuzzy*. Assim como a ART2 (Carpenter; Grossberg, 1987b) e a ART 2-A (Carpenter; Grossberg; Rosen, 1991b), a ART *Fuzzy* também é capaz de aprender a categorizar tanto dados binários quanto analógicos. Entretanto, seu funcionamento é procedente da substituição do operador de interseção da teoria dos conjuntos da ART 1 pela teoria dos conjuntos difusos, por operadores lógicos *Fuzzy* AND/OR. Esta característica torna o algoritmo mais simples em relação aos modelos baseados na ART2 (Carpenter; Grossberg; Rosen, 1991a).

Os operadores lógicos *Fuzzy* AND/OR utilizam o conceito de mínimos e máximos, para lógica *Fuzzy* AND e *Fuzzy* OR, respectivamente e, a partir deles, a classificação da rede

ART *Fuzzy* se dá por hiper-retângulos, de modo que cada categoria j tem uma representação geométrica como um hiper-retângulo R_j , conforme ilustrado na Figura 29.

Figura 29 - Classificação por hiper-retângulo de uma categoria.



Fonte: Adaptado de: (Carpenter; Grossberg; Rosen, 1991c).

Em alguns modelos ART que categorizam entradas analógicas pode ocorrer uma proliferação de categorias. Este fenômeno é evitado no ART *Fuzzy* se as entradas são normalizadas (Moore, 1989). Existem diversos métodos de normalizações de valores, uma delas é a norma da soma:

$$x = \frac{a}{|a|} \quad (14)$$

Tipicamente, as amostras são pré-processadas pela aplicação de codificação complementar. Esta transformação torna o vetor de entrada em 2d-dimensenional, enquanto impõe uma norma constante na forma: $(\|x\|_1 \leftarrow [x, \vec{1} - x])$ (Carpenter *et al.*, 1992; Carpenter; Grossberg; Reynolds, 1991), *i.e.*:

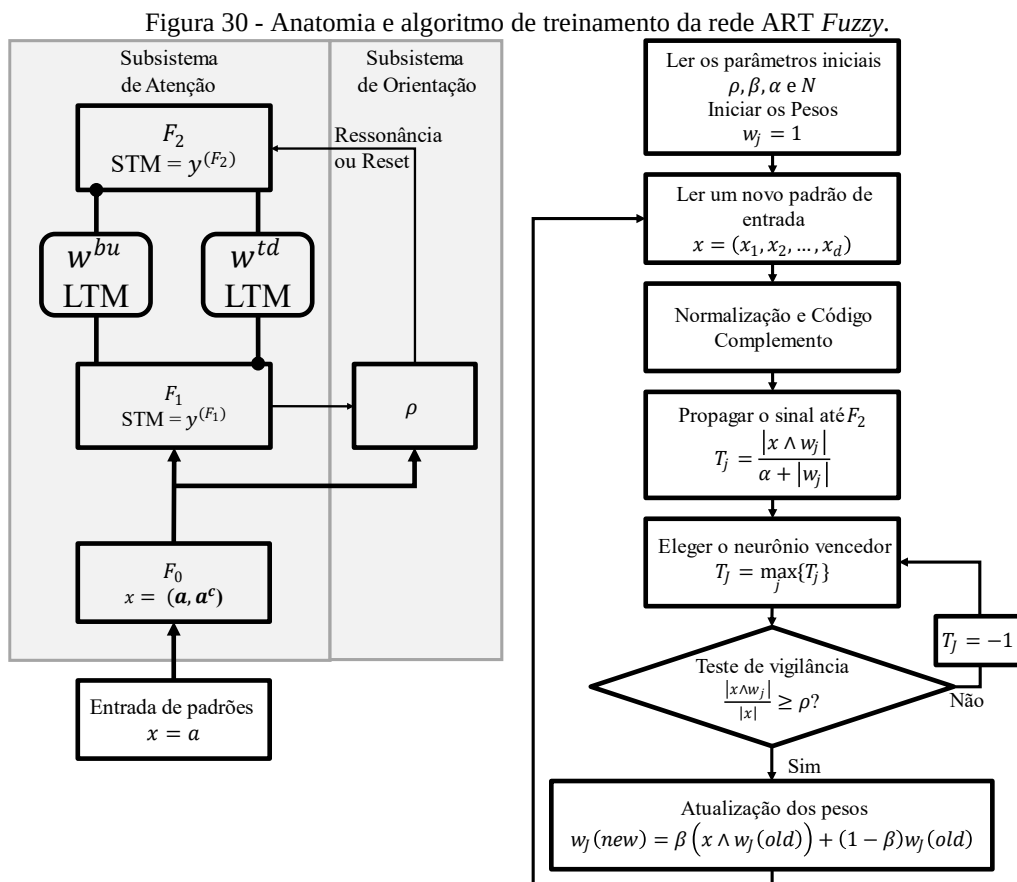
$$\|x\|_1 = \sum_{i=1}^{2d} x_i = \sum_{i=1}^d x_i + \sum_{i=1}^d (1 - x_i) = d \quad (15)$$

Alternativamente, a entrada com o complemento pode ser representada como:

$$x = [a, a^c] = (a_1, a_2, \dots, a_d, 1 - a_1, 1 - a_2, \dots, 1 - a_d) \quad (16)$$

Cada unidade de categoria possui um vetor de peso LTM, portanto, se a codificação de complemento for empregada, então $w_j = [w_j, w_j^c]$. Como consequência de contemplar a etapa de pré-processamento, a ART *Fuzzy* possui uma etapa adicional, a camada F_0 . A anatomia da

rede ART Fuzzy e o fluxograma que representa seu algoritmo de treinamento são ilustrados na Figura 30.



Fonte: Adaptado de: (Carpenter; Grossberg; Rosen, 1991c).

Entre as principais características da ART Fuzzy, são sintetizadas (Carpenter; Grossberg; Rosen, 1991b, 1991c):

- Vetor de entrada: Cada entrada x é um vetor d-dimensional e cada elemento x_j pertencente ao intervalo de 0 a 1. Se a codificação de complemento for empregada, então a entrada se torna 2d-dimensional a partir da camada F_0 .
- Vetor de pesos: Cada categoria j corresponde a um vetor de pesos LTM $w_j = (w_{j1}, w_{j2}, \dots, w_{jd})$ de valores ajustáveis. Se a codificação de complemento for empregada, o vetor de pesos LTM também serão 2d-dimensionais.
- O número de potenciais categorias é arbitrário e cada categoria inativa é composta por pesos $w_j = 1$.
- Definição dos parâmetros: a definição predeterminada dos parâmetros estabelece a dinâmica da ART Fuzzy e são eles:

- Parâmetro de escolha ($\alpha > 0$), que controla a escolha por um menor hiper retângulo (pode ser visto como um parâmetro de regularização que penaliza grandes pesos);
- Parâmetro de vigilância ($\rho \in [0 \ 1]$) define a seletividade do sistema;
- Parâmetro de taxa de treinamento ($\beta \in [0 \ 1]$) – define a velocidade de reajuste dos pesos.

- Operadores lógicos AND/OR são pertencentes à teoria dos conjuntos difusos.

Na etapa de treinamento, o funcionamento da ART *Fuzzy* inicia-se a partir de uma entrada x , que ativa as categorias de cada neurônio j e terá sua atividade determinada pela função escolha:

$$T_j = \frac{|x \wedge w_j|}{\alpha + |w_j|} \quad (17)$$

De modo que $x \wedge y = \min(x_i, y_i)$ e a norma $|\cdot|$ é definida por:

$$|x| = \sum_{i=1}^d x_i \quad (18)$$

por conseguinte, a escolha da categoria se dá por:

$$T_j = \max\{T_j : j = 1, \dots, N\} \quad (19)$$

Quando o nó vencedor J é selecionado, a atividade F_2 é:

$$y_j^{(F_2)} = \begin{cases} 1, & \text{se } j = J \\ 0, & \text{caso contrário} \end{cases} \quad (20)$$

A função de escolha mede o grau em que x é um subconjunto difuso de w_j e tende a categorias menores. A atividade F_1 é definida como:

$$y_j^{(F_1)} = \begin{cases} x, & \text{se } F_2 \text{ está inativo} \\ x \wedge w_j, & \text{caso contrário} \end{cases} \quad (21)$$

Em seguida, o fenômeno da ressonância é verificado, tal que:

$$\frac{|y_j^{(F_1)}|}{|x|} = \frac{|x \wedge w_j|}{|x|} \geq \rho \quad (22)$$

Se a ressonância não for satisfeita, ocorre o resete da categoria J e T_j tem seu valor atualizado para 0 ou -1 , uma nova categoria é escolhida por (19) para buscar uma nova hipótese até que satisfaça a ressonância em (22). Em último caso, uma categoria inativa sempre atenderá o critério de vigilância, pois quando um nó não ativo for recrutado para aprender a amostra x , será inicializado com um vetor de pesos $w = \vec{1}$.

Quando há ressonância, os pesos são atualizados para refinar o conhecimento armazenado pela seguinte relação:

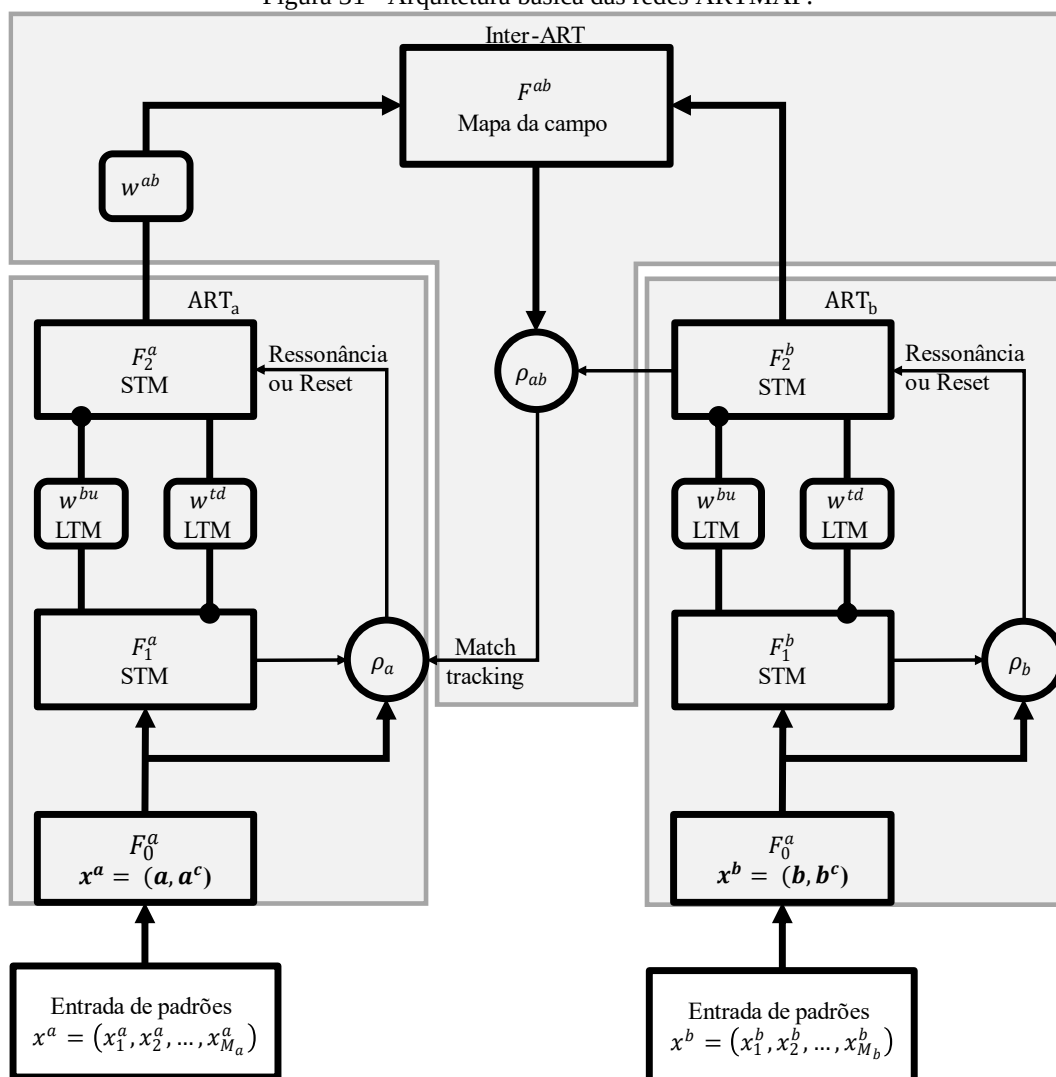
$$w_j(new) = \beta (x \wedge w_j(old)) + (1 - \beta)w_j(old) \quad (23)$$

4.4 ARTMAP E ARTMAP FUZZY

Enquanto uma rede ART mapeia amostras para categorias, uma arquitetura ARTMAP, por treinamento supervisionado, mapeia categorias para classes (Carpenter; Grossberg; Reynolds, 1991). Em relação à arquitetura básica, ilustrada na Figura 31, tanto a ARTMAP quanto ARTMAP *Fuzzy* são constituídas por um par de módulos da teoria da ressonância adaptativa, denominadas de ARTa e ARTb, responsáveis em classificar os padrões de entrada e saída, respectivamente.

Além dos dois módulos, a arquitetura ARTMAP também possui o módulo interART, composto de memórias associativas em seu mapa de campo F^{ab} . Este módulo é crucial, pois, funciona como um subsistema de orientação, *i.e.*, realiza mapeamentos multidimensionais entre as categorias dos dois módulos ART para avaliar a qualidade do mapeamento de ambas, além de verificar a necessidade de adicionar um novo nó com base em um sinal supervisionado.

Figura 31 - Arquitetura básica das redes ARTMAP.



Fonte: Adaptado de: Carpenter *et al.* (1992); Carpenter; Grossberg; Reynolds (1991).

A ARTMAP foi originalmente usada para aprender mapeamentos entre os vetores de entrada e saída binários. A ARTMAP *Fuzzy*, no entanto, consta fundamentalmente da substituição de operadores lógicos AND/OR pelos operadores lógicos *Fuzzy* AND/OR e a sua principal característica, por consequência, é o treinamento supervisionado de padrões de entradas e saídas binários e/ou analógicos (Carpenter *et al.*, 1992; Carpenter; Grossberg; Reynolds, 1991) e, portanto, o fluxograma de treinamento de ambas as redes é similar. Na Figura 32 é apresentado o algoritmo de treinamento da rede ARTMAP *Fuzzy*.

O treinamento inicia-se com todas as categorias inativas e pesos sinápticos com valores unitários de modo a representar memórias em branco, ou seja: $w_{(Nx2M_a)}^a, w_{(Nx2M_b)}^b, w_{(NxN)}^{ab} = 1$. Sendo, M_a, M_b as dimensões das entradas da ARTa e ARTb, respectivamente, e N o número de padrões de entrada apresentadas no treinamento. Além disto, o desempenho de treinamento destas redes depende das especificações definidas pelo

usuário: o parâmetro de escolha (α), dos parâmetros de vigilância dos módulos ARTa, ARTb e inter-ART (respectivamente representados por: ρ_a , ρ_b , ρ_{ab}) e a taxa de treinamento (β).

Definidos os parâmetros e iniciadas as matrizes de pesos, o módulo ARTa recebe parte de um conjunto de padrões de entrada x^a e o módulo ARTb recebe parte de um conjunto de padrões de entrada x^b , tal que x_i^b é o rótulo de x_i^a .

Durante o treinamento, o módulo ARTa está sujeito a um certo nível de concordância com a atividade do ARTb, uma vez que este último codifica os rótulos desejados. Para este procedimento ser realizado, um teste de vigilância adicional é necessário, que usa a resposta da ARTb para acionar uma incompatibilidade ou permitir o aprendizado dada uma previsão incorreta ou correta, respectivamente.

Do mesmo que calculado na Seções 4.2 e 4.3 e, respectivamente, de forma equivalente para ARTMAP e ARTMAP *Fuzzy*, os nós ressonantes para ambos os módulos ART são selecionados após a apresentação de um i -ésimo par de amostra (x_i^a , x_i^b). Em seguida, o rastreamento de correspondência, ou *match tracking*, é calculado pela Equação (24):

$$\frac{|y^{(F_2^b)} \wedge w_j^{ab}|}{|y^{(F_2^b)}|} \geq \rho_{ab} \quad (24)$$

Quando a previsão do ARTa é refutada pelo ARTb, o mapa de campo aciona um mecanismo de rastreamento de correspondência no qual a categoria ressonante do ARTa é inibida, o limiar do parâmetro de vigilância recebe, temporariamente, um incremento de uma quantidade mínima necessária (ε) e o processo de pesquisa é reiniciado, de modo que o ARTa selecione outra categoria ou, se necessário, aprender uma nova categoria cuja predição corresponde à categoria ARTb. Ao envolver este mecanismo de rastreamento de correspondência, o ARTMAP troca generalização por especificidade para diminuir o erro de treinamento.

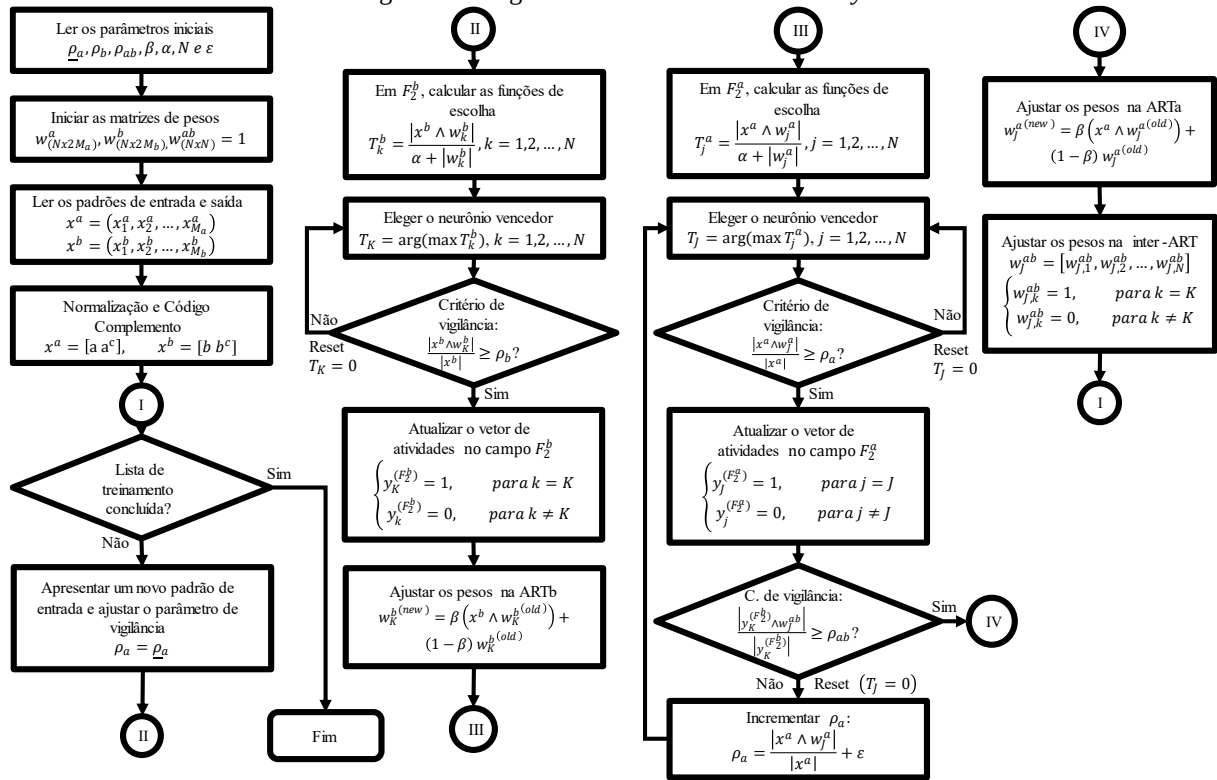
$$\rho_a = \frac{|x^a \wedge w_j^a|}{|x^a|} + \varepsilon \quad (25)$$

Quando finalmente a ressonância entre os dois módulos ocorre, ou seja, quando houver uma associação entre a categoria J de ARTa e a categoria K de ARTb, todos os três módulos aprendem. Os pesos da ARTa e ARTb são atualizados da mesma forma conforme descritos nas seções 4.2 e 4.3 para ARTMAP e ARTMAP *Fuzzy*, respectivamente. No caso da memória associativa LTM de F^{ab} , que é representado por uma matriz $w_{ab} = [w_{ij}^{ab}]_{N_a \times N_b}$,

terá o elemento $w_{jk}^{ab} = 1$ e os demais zerados. Este procedimento é necessário para associar a entrada x_i^a que disparou a categoria J com o rótulo x_i^b que disparou a categoria K .

Após qualquer um dos casos: categoria ressonante encontrada ou uma nova criada; o parâmetro de vigilância da ARTa é novamente relaxado ao seu valor de base ($\rho_a = \underline{\rho}_a$) mediante a apresentação de um novo par de entrada para continuação do processo de treinamento.

Figura 32 - Algoritmo da rede ARTMAP Fuzzy.



Fonte: Adaptado de: Decanini; Tonelli-Neto; Minussi (2012).

No modo de previsão, a saída ARTa é utilizada para fazer uma previsão e recuperar concretamente os rótulos de ARTb por meio da matriz de pesos de F^{ab} . Para isto, basta rastrear o vetor de peso do campo do mapa w_j^{ab} e defini-lo como a saída do sistema, i.e., quando a categoria ressonante J de um ARTa é encontrada, a classe prevista K é obtida pela Equação (26):

$$K = \arg \max_k \left(\sum_{j=1}^{N_a} w_{jk}^{ab} y_j^{(F_2^a)} \right) \quad (26)$$

5 VGG ARTMAP FUZZY

O aprendizado profundo é atualmente muito consolidado e popular ao apresentarem resultados com grande precisão em tarefas de classificação, regressão e agrupamento. Estas aplicações, no entanto, normalmente necessitam de grandes quantidades de dados rotulados e exigem muito tempo de processamento durante a etapa de treinamento e, portanto, são dependentes de alta capacidade computacional. Felizmente, esta dependência pode ser reduzida com o uso do aprendizado por transferência, *i.e.*, ao invés de treinar um modelo de aprendizado profundo do zero, com muitos dados e levar dias de treinamento, é possível reutilizar o conhecimento de um modelo pré-treinado para aplicações em outras tarefas.

Especificamente em visão computacional, redes convolucionais treinadas por grandes conjuntos de dados retêm uma grande quantidade de conhecimento e são capazes de extrair representações de recursos de baixo nível, tais como: espacial, bordas, rotações, iluminações, entre outras. Consequentemente, baseado na transferência de aprendizado, estas redes podem ser reutilizadas como modelos pré-treinados e atuar como extratores de características para diferentes problemas de visão computacional (mesmo em aplicações diferentes do conjunto de dados de origem), possibilitando que novos modelos de rede se beneficiam desta vastidão de dados (Ribani; Marengoni, 2019).

Existem muitas redes convolucionais que alcançaram desempenho de ponta para diferentes tarefas, como o VGG, Inception, ResNet. Esses modelos foram treinados por um longo período usando múltiplas GPUs de alta qualidade a partir de milhões de imagens ordenadas em milhares de categorias. A maioria destes modelos estão disponíveis publicamente e, em muitos casos, os autores compartilham detalhes da implementação, do código-fonte ou até disponibilizam os modelos pré-treinados para *download* e podem ser utilizados como base de estudos de transferência de conhecimento. Muitos destes estão disponibilizados na plataforma Keras - uma API de aprendizado profundo escrita em Python, executada sobre a plataforma de aprendizado de máquina TensorFlow (Abadi *et al.*, 2016).

Este estudo apresenta como proposta a aplicação da arquitetura de aprendizado profundo VGG-16 com pesos pré-treinados para o banco de dados ImageNet, disponibilizado no Keras.

A opção pela arquitetura VGG dentre outras possíveis é fundamentada no bom desempenho para classificação de conjuntos de dados de grande escala (vice-campeã do ILSVRC no ano de 2014) e por apresentar uma estrutura simples e intuitiva, o que proporciona uma excelente capacidade de expansão para outros conjuntos de dados.

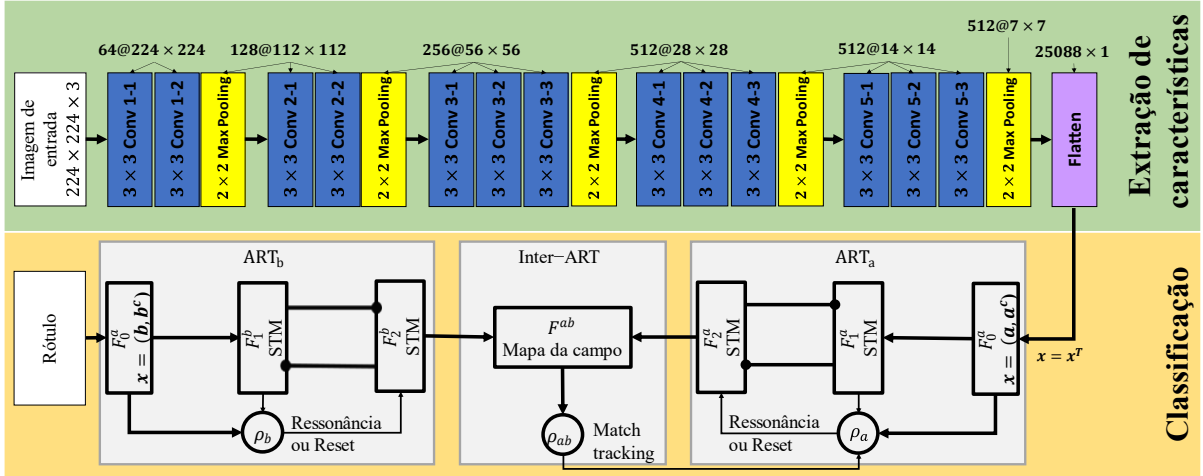
Entretanto, a principal propriedade das redes da família VGG é a enorme quantidade de parâmetros treinados (cerca de 138 milhões na versão VGG-16), o que justifica o uso de forma ampla desta rede em diversos trabalhos como modelo pré-treinado de referência para transferência de aprendizado (Gopalakrishnan *et al.*, 2017; Tammina, 2019; Thepade *et al.*, 2022; Yang *et al.*, 2020; Ye *et al.*, 2019).

Esta enorme quantidade de parâmetros treinados permite que este modelo extraia recursos de imagens com grande sensibilidade, viabilizando o classificador a distinguir com maior facilidade uma imagem de outra entre as classes possíveis.

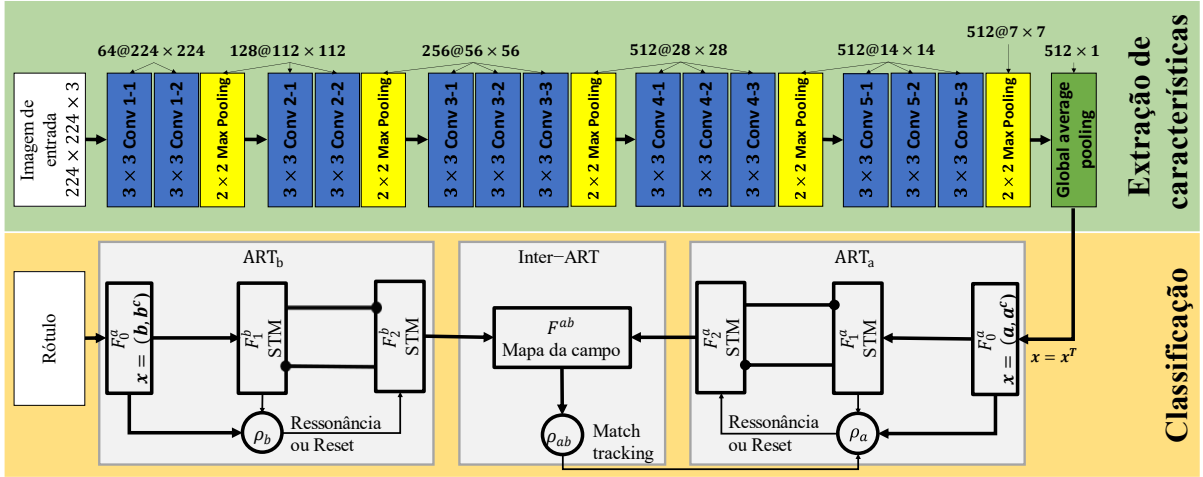
A partir das características extraídas em imagens pela CNN, as redes baseadas na Teoria da Ressonância Adaptativa apresentam fundamentalmente características promissoras se utilizadas como classificadores, uma vez que possuem sistemas tanto atencional quanto orientador e realizam ajustes dos pesos sinápticos de forma local e, portanto, respeitam o dilema da plasticidade e estabilidade. Adicionalmente - e em contrapartida às redes baseadas no algoritmo de retropropagação que podem apresentar esquecimento catastrófico - as redes respaldadas nos conceitos presentes da família ART permitem treinamentos futuros sem comprometer o conhecimento anteriormente adquiridos.

A arquitetura proposta, denominada de VGG ARTMAP *Fuzzy*, é apresentada em duas variantes, representadas na Figura 33 e Figura 34, com a utilização de camadas de vetor resultante após procedimento de achatamento (*flattening*) e *pooling* médio global, respectivamente. Em ambas as versões, apenas os cinco blocos convolucionais da VGG-16 pré-treinada foram utilizados e são responsáveis pelas extrações de características. Em função da abordagem de transferência de conhecimento utilizada (representação de características), os pesos das camadas convolucionais são mantidos congelados. Por outro lado, as camadas totalmente conectadas originais da VGG-16 treinadas para o banco de dados ImageNet foram removidas e substituídas por uma rede ARTMAP *Fuzzy* para a tarefa de classificação.

Figura 33 - Arquitetura VGG-ARTMAP Fuzzy: versão com achatamento.



Fonte: Elaboração do próprio autor.

Figura 34 - Arquitetura VGG-ARTMAP Fuzzy: versão com *pooling* médio global.

Fonte: Elaboração do próprio autor.

Como propriedade da VGG-16 treinada para a ImageNet, em sua entrada são apresentadas imagens RGB de dimensões 224×224 , ou seja, $224 \times 224 \times 3$. Além disto, é realizado um pré-processamento de subtrair o valor RGB médio de cada pixel (Simonyan; Zisserman, 2014). Caso a imagem de entrada não possua as dimensões adequadas, é realizado um redimensionamento por uma interpolação bilinear como um ajuste prévio.

As etapas de extração de características e classificação do modelo trabalham de forma síncrona, *i.e.*, para cada imagem apresentada na entrada da VGG-16, o quinto bloco convolucional disponibiliza 512 mapas de características de dimensões 7×7 que são utilizados na classificação pela ARTMAP Fuzzy. Em ambas as versões, o vetor disponibilizado na etapa de extração de características é transposto, visto que, o padrão de entrada da ARTMAP Fuzzy deve estar no formato vetor linha.

Na versão de achatamento, todos os coeficientes dos mapas de características são reorganizados para uma camada de vetor único. Nesta condição, a VGG-16 disponibiliza em

sua saída um vetor de dimensão 25088×1 que é transposto antes de ser apresentado à camada F_0^a da ARTMAP *Fuzzy*.

Já na segunda versão, estes mapas são sintetizados por uma camada de *pooling* médio global (Lin; Chen; Yan, 2013) que compacta as informações de cada mapa de características em um único valor médio, portanto, disponibiliza um vetor de dimensão 512×1 que, da mesma forma da outra versão, é transposto antes de ser apresentado à ARTMAP *Fuzzy*. A opção pela camada de *pooling* médio global em relação a operação de achatamento tem a intenção de reduzir os parâmetros apresentados para a ARTMAP *Fuzzy*, sem comprometer a sensibilidade da extração de características.

Antes de serem apresentados à camada F_0^b da ARTMAP *Fuzzy*, os rótulos são pré-processados pela técnica *One-Hot-Encoding*. Nesse tipo de codificação, cada rótulo é representado por um grupo de bits com apenas um bit alto (1) e todos os outros baixos (0), além disso, a quantidade de bits necessárias para representar os rótulos é o número de rótulos exclusivos disponíveis no conjunto original. Na Tabela 6 é ilustrado um exemplo de pré-processamento por *One-Hot-Encoding* a partir de um conjunto de dados composto por quatro rótulos possíveis.

Tabela 5 - Exemplo de processamento por *One-Hot-Encoding*.

Rótulo original	<i>One-Hot-Encoding</i>
Rótulo 0	0001
Rótulo 1	0010
Rótulo 2	0100
Rótulo 3	1000
Rótulo 2	0100
Rótulo 1	0010

Fonte: Elaboração do próprio autor.

Todo o processamento da ARTMAP *Fuzzy*, incluído pré-processamento por código complemento, segue o equacionamento e fluxograma apresentados na Seção 4.4.

6 SIMULAÇÕES E RESULTADOS

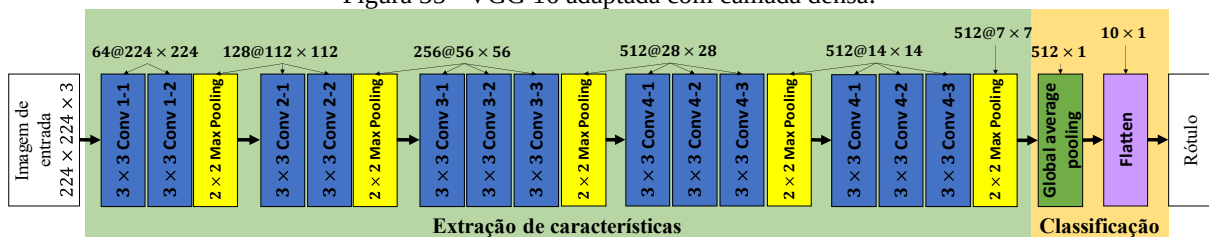
Este capítulo apresenta os bancos de dados utilizados para simulações da arquitetura VGG-ARTMAP *Fuzzy*, os dados do desempenho durante as fases de treinamento assim como a obtenção da acurácia obtida a partir do conjunto de testes. Adicionalmente, como o tempo de processamento é um assunto de interesse na área de visão computacional, os tempos de durações que o algoritmo atuou nas fases de treinamento e teste também foram considerados nos resultados. Destaca-se que, para cada conjunto de dados, serão apresentados apenas os dados de ensaios que alcançaram melhor acurácia.

Com o objetivo de comparação, além das duas variantes propostas (Figura 33 e Figura 34), foi utilizada uma terceira rede neural: uma rede convolucional baseada na VGG-16. Do mesmo modo que as variantes propostas com os classificadores baseados na ART, as camadas convolucionais desta rede, treinadas para o banco de dados Imagenet, tiveram os pesos congelados.

A etapa de classificação desta rede é composta por uma camada de *pooling* médio global seguida de uma camada densa - obtida a partir de um processo de achatamento (*flatten*) - com o número de neurônios equivalentes ao número de rótulos disponíveis no banco de dados utilizados nas simulações. Sendo assim, apresenta-se o modelo na Figura 35, o qual é aplicado para o banco de dados MNIST que necessita de dez neurônios na camada de saída para realizar a classificação de todas as classes e, neste caso, dispõe de um total de 5130 parâmetros treináveis.

Essa rede também é composta pela não linearidade *softmax* na camada densa - recomendada para aplicações de classificação - do otimizador Adam e da função de perda entropia cruzada binária (*Binary Cross-Entropy*)(Kingma; Ba, 2014).

Figura 35 - VGG 16 adaptada com camada densa.



Fonte: Elaboração do próprio autor.

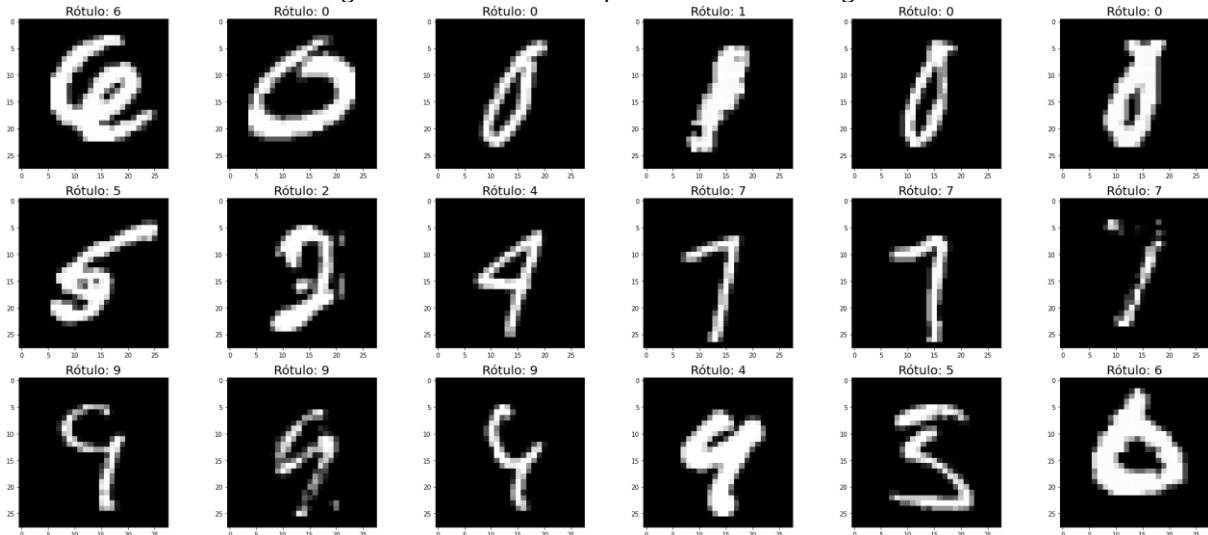
A implementação e as simulações dos algoritmos foram realizadas em linguagem Python, versão 3.10.13 em 64-bit, pela IDE Spyder, em um computador pessoal com uma CPU Intel(R) Core(TM) i7-10700K @ 3.80GHz, 32GB de RAM e equipado com uma GPU NVIDIA GeForce RTX 3070.

6.1 MNIST

O MNIST (*Modified National Institute of Standards and Technology*) é um conjunto de dados desenvolvido por LeCun; Cortes; Burges (2010) que se tornou referência para avaliar modelos de aprendizado de máquina para o problema de classificação de dígitos manuscritos. As imagens com pixels em tons de cinza apresentam dimensões de 28×28 pixels (784 pixels no total), e possuem dígitos únicos centralizados que variam na faixa de 0 a 9, retirados de vários documentos digitalizados. Em seu formato padrão, o conjunto de dados é composto por 60.000 imagens separadas para o treinamento e mais 10.000 para a testes.

Por serem dígitos manuscritos, muitos destes são ambíguos ou de difícil classificação ao olho humano, como as amostras representadas na Figura 36.

Figura 36 - MNIST: Exemplos de amostras ambíguas.



Fonte: Adaptado de: LeCun; Cortes; Burges (2010).

Para a adequação com a entrada da rede para o padrão ImageNet, as imagens foram redimensionadas para 224×224 pixels, replicados para mais dois canais, *i.e.*, adequando para a dimensão $224 \times 224 \times 3$. Esse pré-processamento consome muita memória e, consequentemente, por limitações, nos testes seguintes foram utilizadas apenas 5000 imagens do conjunto MNIST, sendo 80% das imagens separadas de forma aleatória para o treinamento e os outros 20% foram utilizadas em teste para obtenção da acurácia.

Nas Seções 6.1.1, 6.1.2 e 6.1.3 são apresentados os desempenhos dos modelos em relação ao conjunto MNIST que são posteriormente explanados e discutidos na Seção 6.1.4.

6.1.1 VGG ARTMAP Fuzzy com achatamento

O treinamento da rede VGG ARTMAP Fuzzy com achatamento foi concluído após 3 dias e 18 horas, enquanto a fase de teste durou 1 dia e 3 horas. O algoritmo alcançou uma

acurácia de 94,3%. Na Tabela 5 são apresentadas outras informações relacionadas ao desempenho do modelo para este banco de dados.

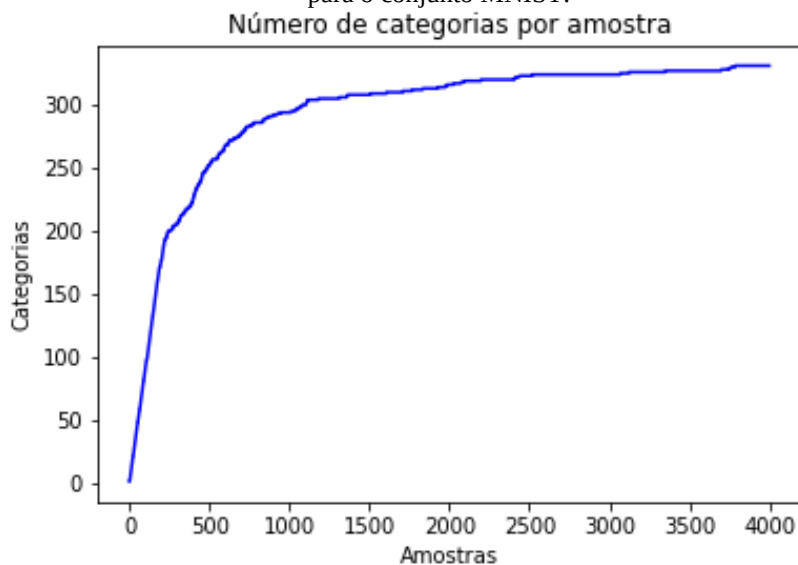
Tabela 6 - Dados de treinamento e teste do conjunto MNIST - VGG ARTMAP *Fuzzy* com achatamento.

Dados	Valor
Parâmetro de vigilância da ARTa (ρ_a)	0,74
Parâmetro de vigilância da ARTb (ρ_b)	0,99
Parâmetro de vigilância do módulo Inter-ART (ρ_{ab})	0,71
Parâmetro de escolha (α)	0,01
Taxa de treinamento (β)	1
Incremento da taxa de treinamento (ε)	0,03
Tamanho do conjunto de treinamento	4000 imagens
Tempo de treinamento (dia, horas:minutos:segundos)	3, 18:00:55
Tamanho do conjunto de teste	1000 imagens
Número de categorias da ARTa	331
Número de categorias da ARTb	10
Tempo de teste (dia, horas:minutos:segundos)	1, 3:01:05
Acurácia	94,3%

Fonte: Elaboração do próprio autor.

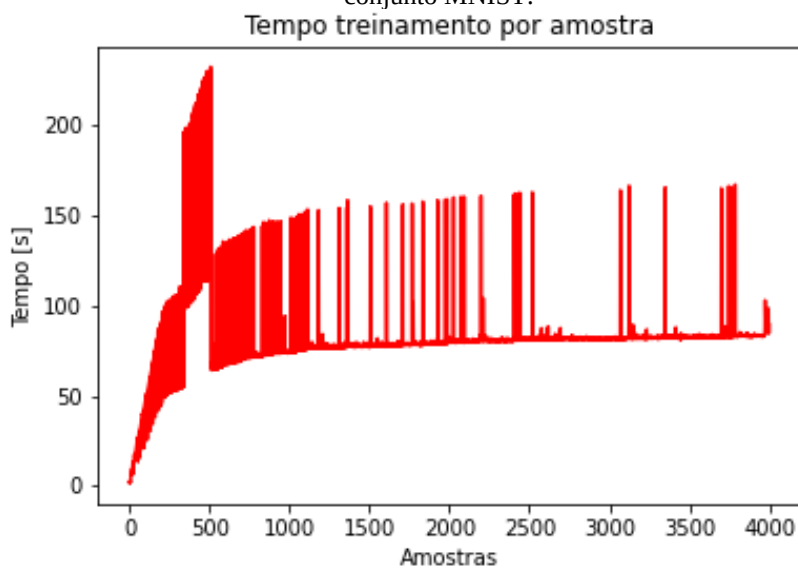
Como a ARTMAP *Fuzzy* é uma rede incremental, a relação e quantidade de categorias criadas representa o comportamento da rede durante a fase de treinamento. Adicionalmente, o poder computacional exigido é diretamente proporcional ao número de categorias criadas. O módulo ARTb normalmente não precisa criar muitas categorias, pois tem a simples tarefa de classificação, uma vez que os rótulos são pré-processados para o tipo *One-hot-encoding*. Por outro lado, o módulo ARTa é responsável por categorizar, a partir da melhor correspondência, o vetor de características disponibilizado pela etapa de extração da VGG-16 e, portanto, o comportamento deste módulo durante a etapa de treinamento é estudado com maior detalhamento. Na Figura 37 é apresentado o comportamento incremental da rede ARTMAP *Fuzzy* com a relação do número de categorias da rede ARTa por amostra treinada e na Figura 38 é representado o tempo de treinamento para cada amostra.

Figura 37 - VGG ARTMAP *Fuzzy* com achatamento - relação do número de categorias da ARTa por amostra para o conjunto MNIST.



Fonte: Elaboração do próprio autor.

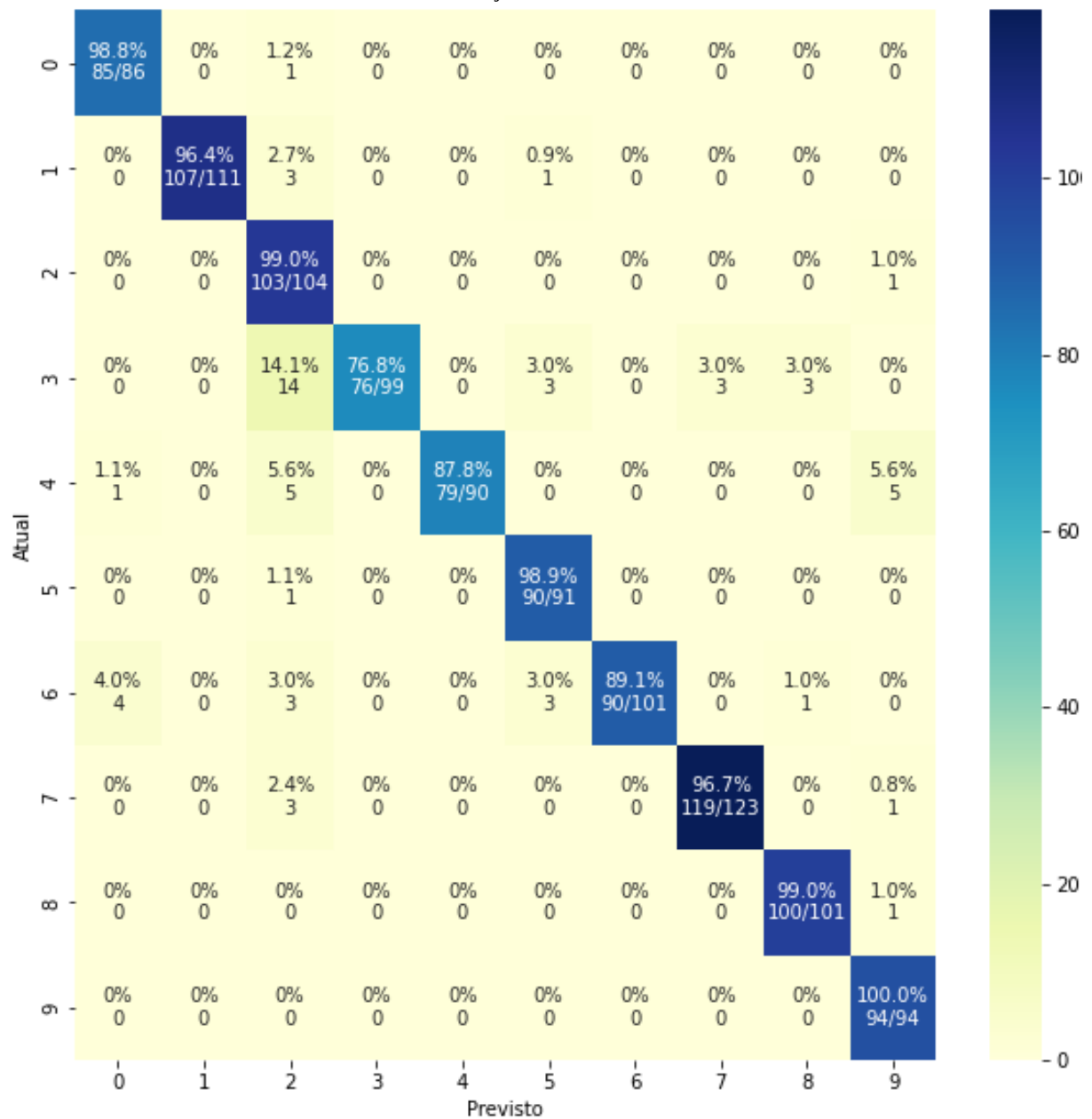
Figura 38 - VGG ARTMAP *Fuzzy* com achatamento - relação do tempo de treinamento por amostra para o conjunto MNIST.



Fonte: Elaboração do próprio autor.

Na Figura 39 é apresentada a matriz de confusão que apresenta o desempenho do sistema na etapa de teste.

Figura 39 - VGG ARTMAP *Fuzzy* com achatamento - Desempenho do sistema para a etapa de teste com o conjunto MNIST.



Fonte: Elaboração do próprio autor.

6.1.2 VGG ARTMAP *Fuzzy* com *pooling* médio global

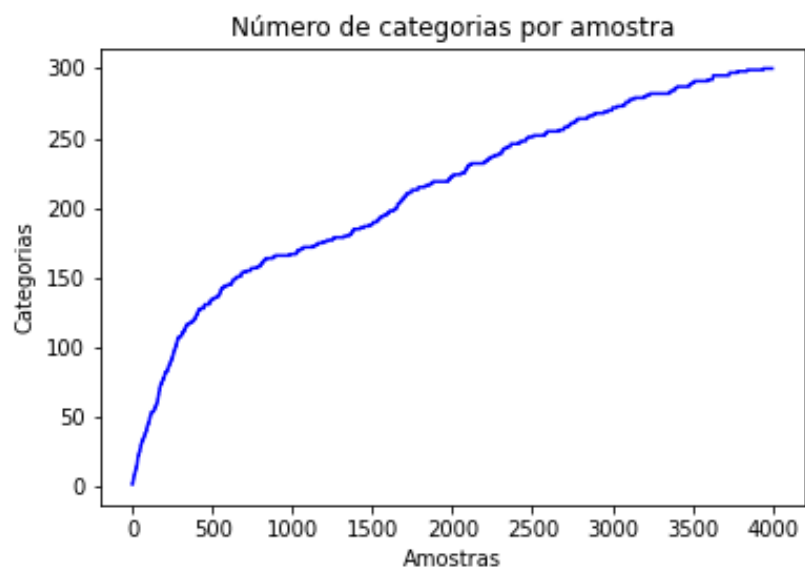
Por sintetizar o vetor de saída da VGG-16, a rede VGG ARTMAP *Fuzzy* com *pooling* médio global apresentou melhor velocidade de processamento, porém menor acurácia nos testes. O treinamento foi concluído após 1 hora e 23 minutos e, após quase 27 minutos de teste, foi constatada uma acurácia de 77%. Do mesmo modo, na Tabela 7 são apresentadas as informações do modelo, na Figura 40, Figura 41 e Figura 42 são apresentados, respectivamente, a relação do número de categorias da ARTa por amostra treinada, o tempo de treinamento por amostra treinada e a matriz de confusão obtido após a etapa de teste.

Tabela 7 - Dados de treinamento e teste do conjunto MNIST - VGG ARTMAP *Fuzzy* com *pooling* médio global.

Dados	Valor
Parâmetro de vigilância da ARTa (ρ_a)	0,74
Parâmetro de vigilância da ARTb (ρ_b)	0,99
Parâmetro de vigilância do módulo Inter-ART (ρ_{ab})	0,71
Parâmetro de escolha (α)	0,01
Taxa de treinamento (β)	1
Incremento da taxa de treinamento (ε)	0,03
Tamanho do conjunto de treinamento	4000 imagens
Tempo de treinamento (dia, horas:minutos:segundos)	1:23:02
Tamanho do conjunto de teste	1000 imagens
Número de categorias da ARTa	300
Número de categorias da ARTb	10
Tempo de teste (dia, horas:minutos:segundos)	0:26:55
Acurácia	77,0%

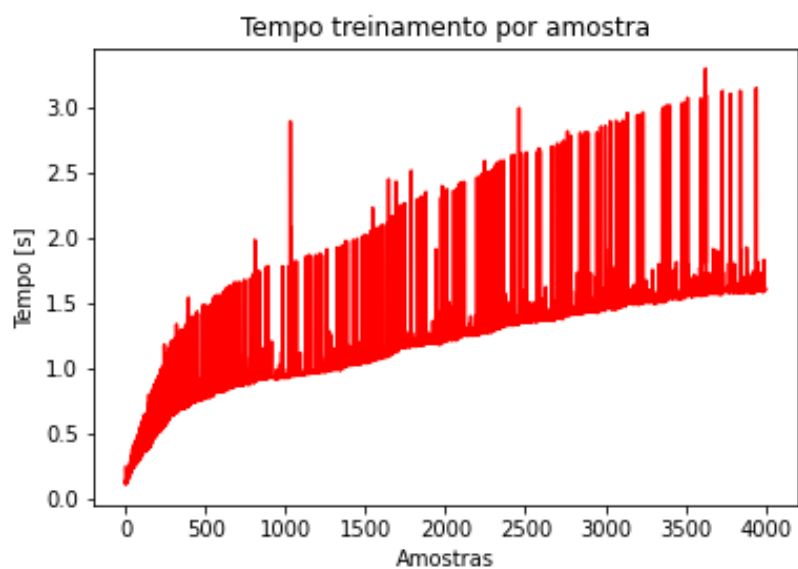
Fonte: Elaboração do próprio autor.

Figura 40 - VGG ARTMAP *Fuzzy* com *pooling* médio global - relação do número de categorias da ARTa por amostra para o conjunto MNIST.



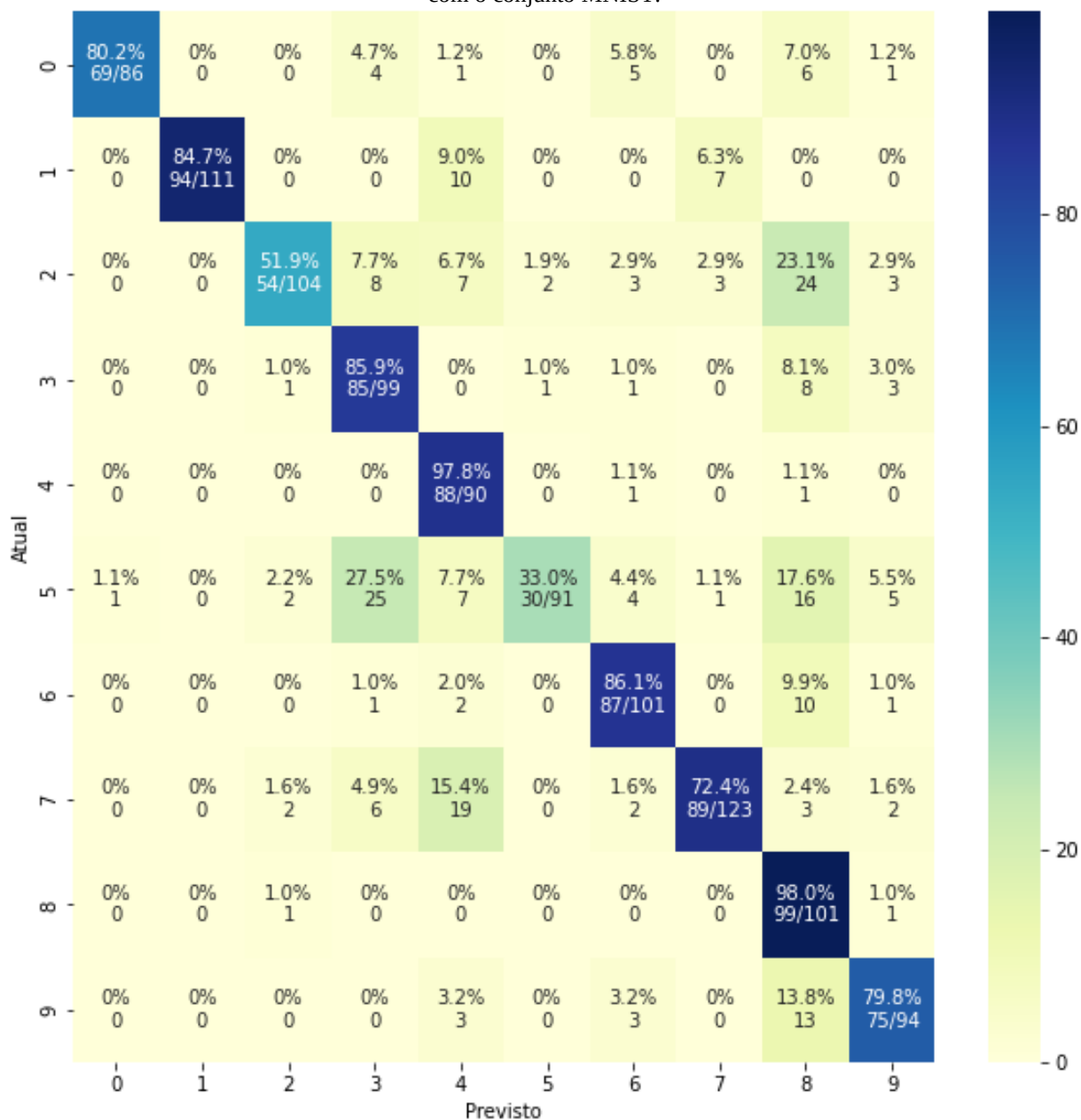
Fonte: Elaboração do próprio autor.

Figura 41 - VGG ARTMAP *Fuzzy* com *pooling* médio global - relação do tempo de treinamento por amostra para o conjunto MNIST.



Fonte: Elaboração do próprio autor.

Figura 42 - VGG ARTMAP *Fuzzy* com *pooling* médio global - Desempenho do sistema para a etapa de teste com o conjunto MNIST.



Fonte: Elaboração do próprio autor.

6.1.3 VGG-16 adaptada

Foram utilizadas 10 épocas para o treinamento de versão adaptada da VGG-16 (ilustrada na Figura 35) para classificar o conjunto MNIST. O algoritmo necessitou de aproximadamente 2 minutos para concluir o treinamento e na fase de testes apresentou uma acurácia de 96,4%.

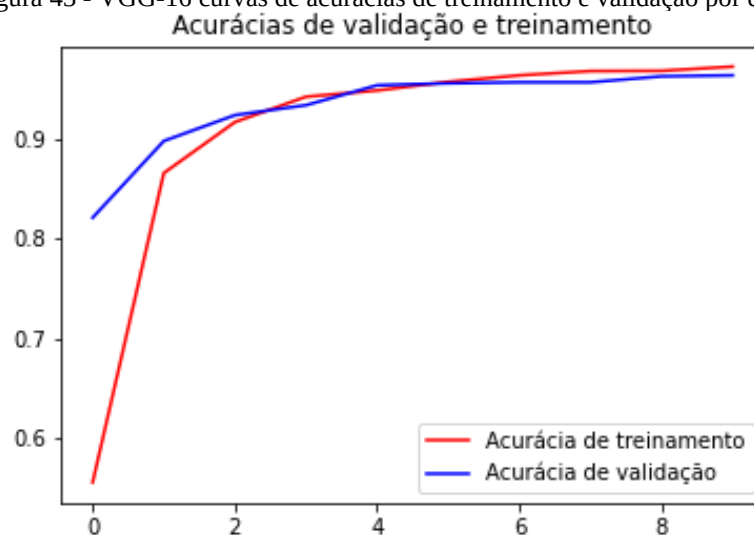
Tabela 8 - Dados de treinamento e teste do conjunto MNIST - VGG adaptada.

Dados	Valor
Tamanho do conjunto de treinamento	4000 imagens
Tempo de treinamento (horas:minutos:segundos)	0:01:57
Tamanho do conjunto de teste	1000 imagens
Acurácia	96,4%

Fonte: Elaboração do próprio autor.

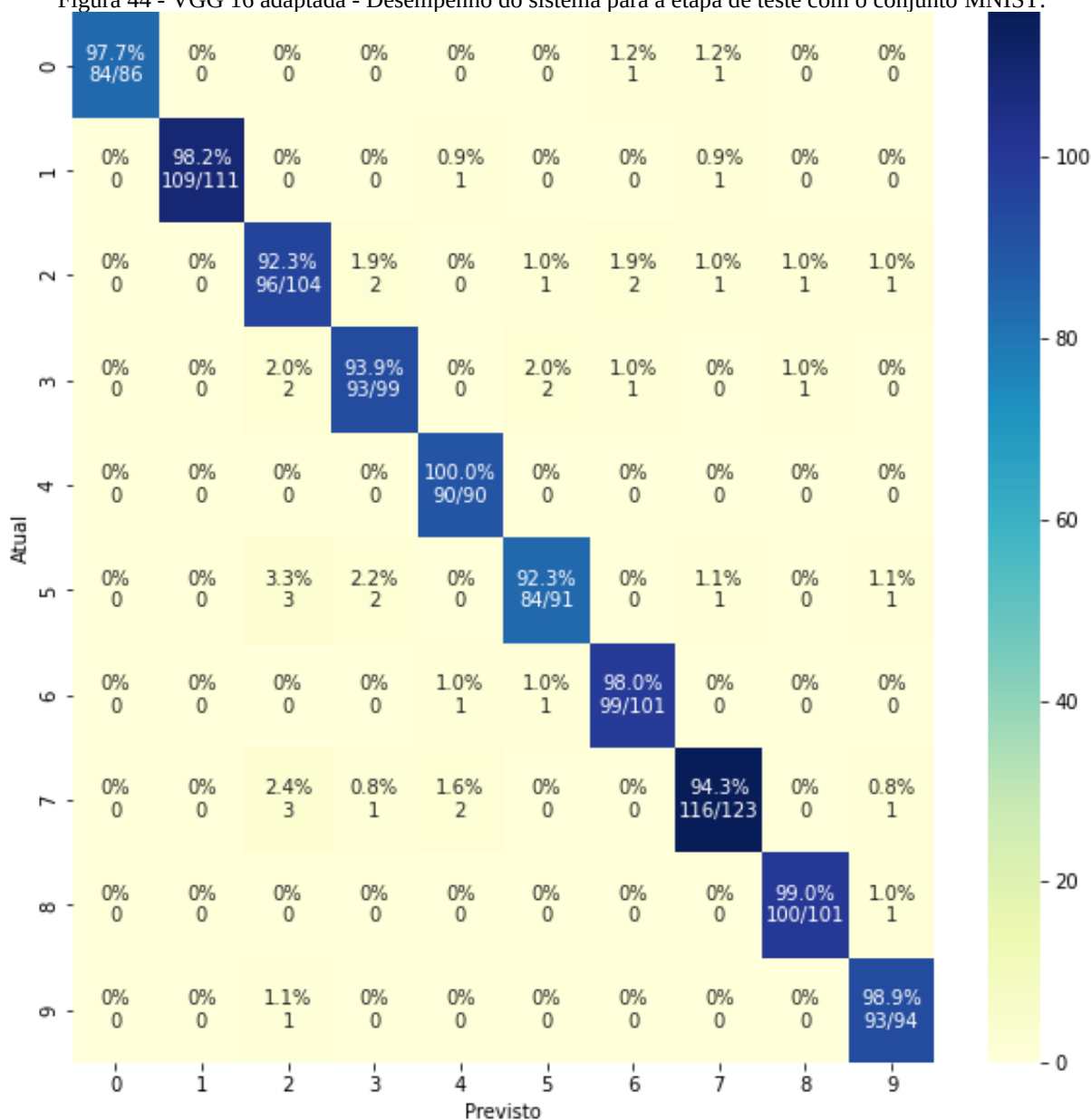
Na Figura 43 são representadas as curvas de desempenho da rede VGG-16 por época e na Figura 44 o mapa de respostas obtidas na etapa de teste.

Figura 43 - VGG-16 curvas de acurácias de treinamento e validação por época.



Fonte: Elaboração do próprio autor.

Figura 44 - VGG 16 adaptada - Desempenho do sistema para a etapa de teste com o conjunto MNIST.



Fonte: Elaboração do próprio autor.

6.1.4 Considerações sobre os resultados: MNIST

O MNIST é um conjunto de dados “termômetro” na área de visão computacional. A partir dele é possível validar o potencial de algoritmos de Inteligência Artificial. Dito isto, ambas variantes propostas encerram o treinamento com o módulo ARTa com a dimensão em torno de 300 categorias. No entanto, a curva de relação do número de categorias por amostra da variante com achatamento (Figura 37) apresentou comportamento de convergência. Isto significa que, na maioria dos casos, as categorias já aprendidas forneciam compatibilidade suficiente para as novas amostras apresentadas no decorrer do treinamento.

Por outro lado, a variante com *pooling* médio global (Figura 40) ainda apresentava uma curva crescente de categorias, ou seja, o treinamento foi interrompido precocemente e o

módulo ARTa ainda precisava de mais experiências de treinamento para melhor representar as amostras futuras.

O comportamento de picos nas curvas de relação do tempo de treinamento por amostra (Figura 38 e Figura 41) acontece quando, naquela etapa de treinamento, a ARTMAP *Fuzzy* não correlaciona o padrão de entrada com as categorias anteriormente aprendidas e, logo, cria-se uma categoria. Neste caso, o processo de treinamento da amostra demanda maior tempo, visto que, uma nova categoria é criada apenas depois de testar todas as hipóteses de categoria vencedora, armazenadas nos módulos ARTs, e as suas respectivas ressonâncias forem menores do que o limite mínimo denominado de parâmetro de vigilância.

Como supracitado, por limitação de memória e proporcionar uma comparação justa, os treinamentos foram padronizados em 5000 amostras, e não as 60000 disponibilizadas pelo banco de dados na sua forma padrão. Entretanto, a variante com *pooling* médio global exige menor poder computacional e, logo, possui potencial de melhora da acurácia a partir de futuras simulações com mais amostras do conjunto MNIST.

A versão da VGG-16 adaptada, apresentou um comportamento de rendimento computacional muito acima dos modelos VGG ARTMAP *Fuzzy* propostos. Essa diferença em relação ao tempo de processamento computacional já era esperada, dado que o algoritmo da VGG-16 opera de forma computacionalmente otimizada por processamento paralelo em GPU. Por outro lado, em relação a acurácia, as curvas ilustradas na Figura 43 indicam um comportamento de saturação no incremento do aprendizado por época. Evidentemente, como o tempo de treinamento foi muito abaixo em relação aos outros modelos, esta rede possivelmente apresentaria um melhor acurácia à medida que o treinamento continuasse. Ainda que, o *overfitting* pudesse ocorrer.

Na Tabela 9 é ilustrado o resumo das acurácias das variantes e VGG-16 adaptada para o treinamento de 5000 amostras do banco de dados MNIST.

Tabela 9 - Comparação de acurácias para MNIST.

	VGG ARTMAP <i>Fuzzy</i> com achatamento	VGG ARTMAP <i>Fuzzy</i> com <i>pooling</i> médio global	VGG-16 adaptada
ACURÁCIA	94,3%	77,0%	96,4%

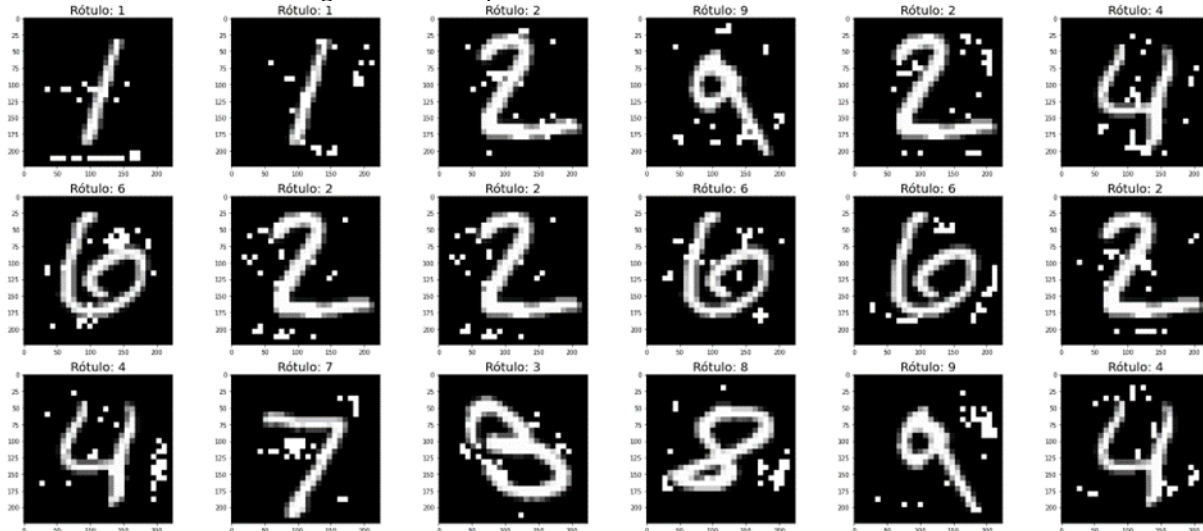
Fonte: Elaboração do próprio autor.

6.2 MNIST ADVERSÁRIO

Para analisar o desempenho das redes neurais quanto à vulnerabilidade a ataques adversários, foi utilizado um banco de dados composto por 90 imagens do conjunto MNIST

somadas com ruído adversário do tipo JSMA (*Jacobian-based Saliency Map Attack, One Pixel*)(Papernot *et al.*, 2016). Na Figura 45 são ilustrados exemplos de imagens com esses ruídos e nas Seções 6.2.1 6.2.2 são apresentados os desempenhos dos modelos e na Seção 6.2.3 são apostadas as considerações.

Figura 45 - Exemplos de amostras adversárias do MNIST.

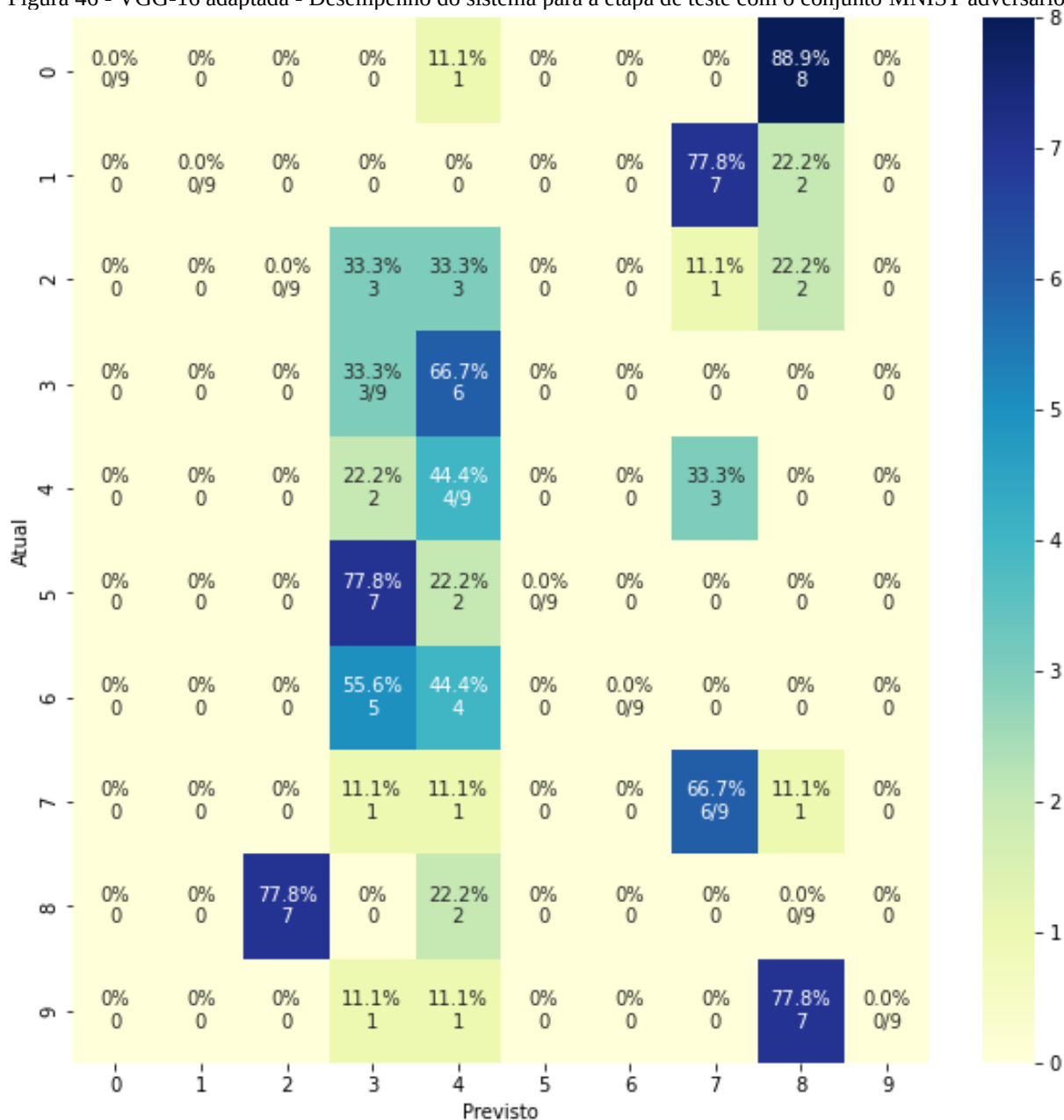


Fonte: Elaboração do próprio autor.

6.2.1 VGG-16 adaptada

Foi aproveitada a VGG-16 supracitada no 6.1.3 - treinada para o conjunto MNIST - para realizar os testes com amostras do MNIST com ruídos. Após apresentar as 90 imagens do conjunto de teste, a VGG-16 apresentou acurácia de 14,44%. Na Figura 46 é ilustrado o mapeamento das respostas da rede.

Figura 46 - VGG-16 adaptada - Desempenho do sistema para a etapa de teste com o conjunto MNIST adversário.

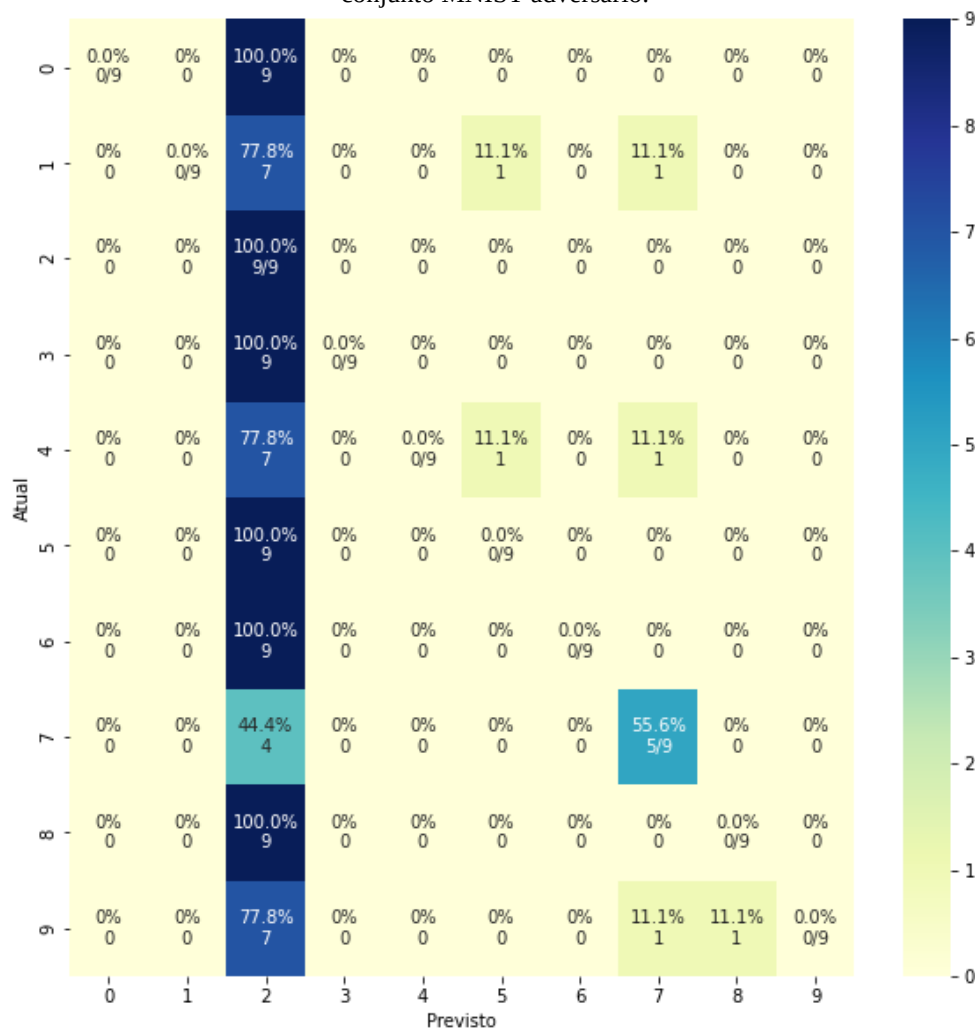


Fonte: Elaboração do próprio autor.

6.2.2 VGG ARTMAP Fuzzy com achatamento

Para testar o desempenho da rede VGG ARTMAP *Fuzzy* foi utilizada a variante com achatamento, abordada na Seção 6.1.1 treinada para o conjunto MNIST. Após validação das 90 imagens, o conjunto apresentou uma acurácia de 15,55%. Na Figura 47 apresenta-se o mapeamento das respostas da rede.

Figura 47 - VGG ARTMAP *Fuzzy* com achatamento - Desempenho do sistema para a etapa de teste com o conjunto MNIST adversário.



6.2.3 Considerações sobre os resultados: MNIST adversário

Após as simulações supracitadas, é perceptível que a VGG-16 adaptada e a VGG ARTMAP *Fuzzy*, na versão com achatamento, que apresentaram acurácias acima de 90% no que tange o conjunto MNIST sem ruídos (Figura 36), não conseguiram identificar as amostras do conjunto com ruídos causados por ataques adversários (Figura 45); imagens que, apesar dos ruídos, são possíveis de serem reconhecidas ao olho humano.

As acurácias de aproximadamente 14% e 15%, apresentadas nas Seções 6.2.1 e 6.2.2, representa acertos aleatórios e as redes se comportam como se estivessem sendo expostas para um novo conjunto de dados, diferente do que elas foram treinadas. A proposta VGG ARTMAP *Fuzzy*, também não foi capaz de classificar corretamente os exemplos adversários.

6.3 OUTROS BANCOS DE DADOS

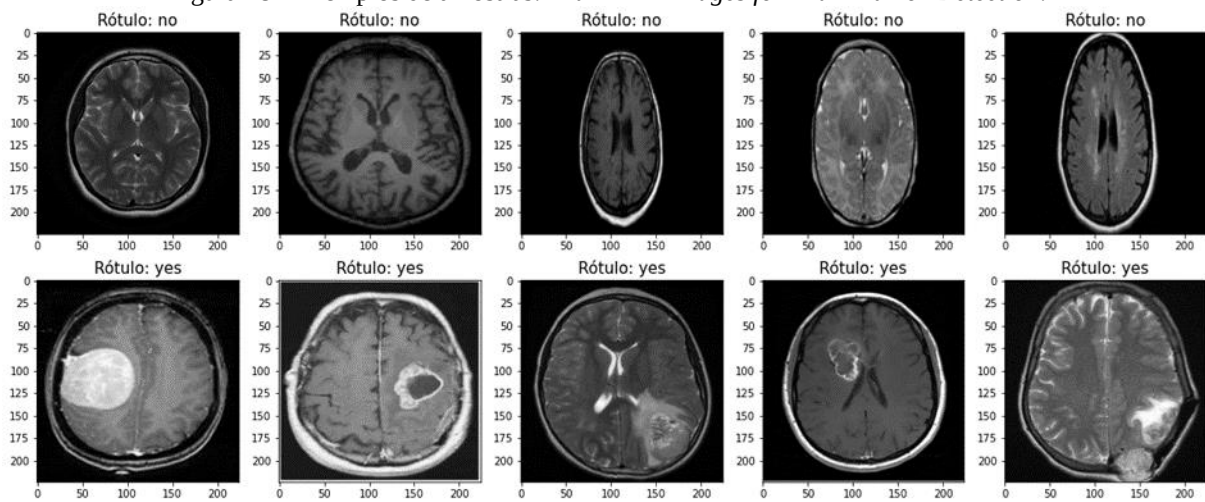
Na maioria dos casos, as técnicas de ANN são utilizadas para aplicações próprias, com bancos de dados pequenos e específicos. Portanto, para validar a versatilidade da proposta, o modelo VGG ARTMAP *Fuzzy* com *pooling* médio global foi utilizado para testes com bancos de dados variados que simulam aplicações em diversas áreas, tais como: saúde, agronomia e transporte.

Os bancos de dados apresentados entre as Seções 6.3.1 e 6.3.6 foram retirados na plataforma Kaggle e separados de forma aleatória com proporção de 80% de amostras para treino e 20% para testes. Na seção 6.3.7 são apresentados os comentários e as acurácias do modelo VGG ARTMAP *Fuzzy* com *pooling* médio global, comparadas com as acurácias da rede VGG-16 adaptada.

6.3.1 Brain MRI Images for Brain Tumor Detection

Este banco de dados é composto por 253 imagens de ressonância magnética para detecção de tumor cerebral. Portanto, os rótulos das classes são dois possíveis: Sim ou Não, conforme ilustrado na Figura 48. As classes são separadas em 155 imagens com o rótulo “Yes” e 98 como “No” (Navoneel Chakrabarty; Kaggle, 2019).

Figura 48 - Exemplos de amostras: *Brain MRI Images for Brain Tumor Detection*.



Fonte: Elaboração do próprio autor.

Originalmente, as imagens em formato “.jpg” e possuem diversas dimensões, 3 canais e, portanto, são previamente redimensionadas para estar em conformidade ao padrão ImageNet pré-treinada pela VGG-16.

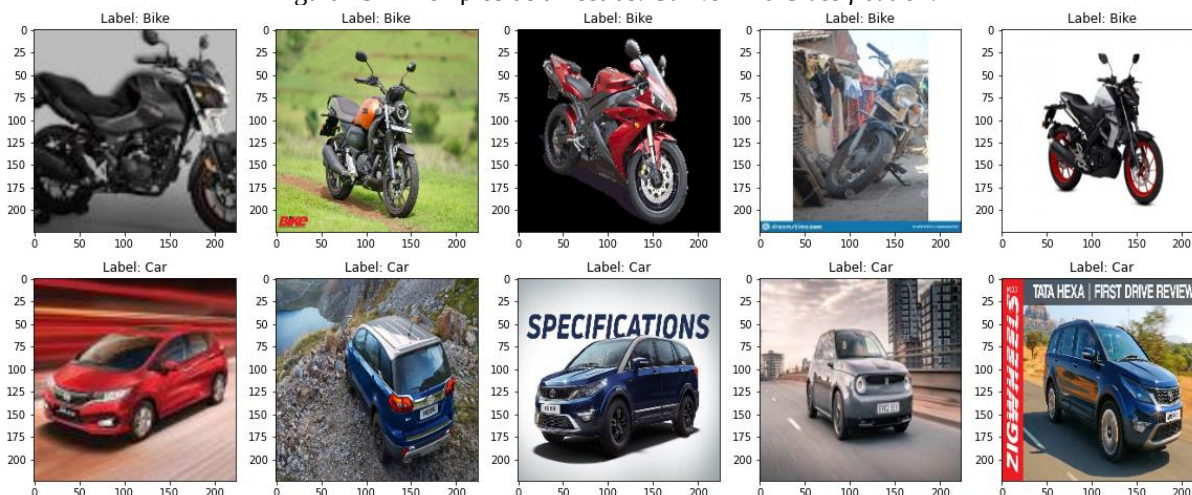
6.3.2 Car vs Bike Classification

Este conjunto de dados é uma coleção de um total de 4.000 imagens divididas proporcionalmente entre motos e carros, ou seja, 2000 imagens da classe “Car” e 2000 da classe “Bike” (DeepNets; Kaggle, 2022).

As imagens estão disponibilizadas em diversas dimensões, em 3 canais e em diversos formatos, como: “.jpeg”, “.jpg” e “.png”. Portanto, esse banco de dados é redimensionado para o padrão ImageNet.

Este banco de dados possui uma grande diversidade de carros e motos, em diferentes tipos de ambientes e, até mesmo sem, ambiente algum (imagens sem fundo); imagens de interiores de veículos, ou partes deles e; portanto, este conjunto obriga o modelo a entender os diferentes subconjuntos em cada classe para apresentar um bom desempenho.

Figura 49 - Exemplos de amostras: Car vs Bike Classification.

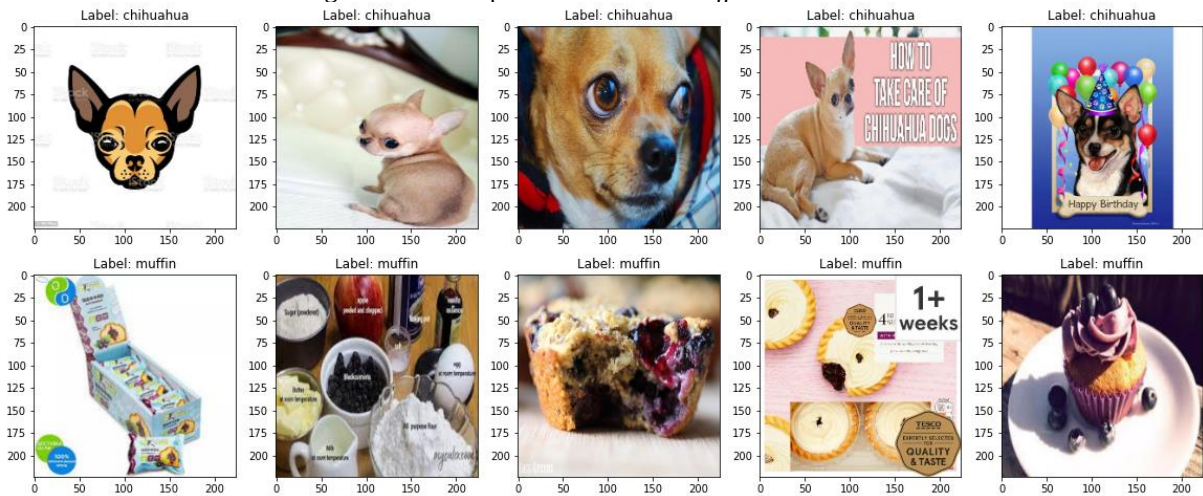


Fonte: Elaboração do próprio autor.

6.3.3 Muffin vs chihuahua

Este conjunto de dados propõe um problema de classificação de imagens a partir da ambiguidade visual entre *muffins* e *chihuahuas*. De acordo com o colaborador as imagens foram extraídas do Google Images e, na versão utilizada, 5917 imagens são totalizadas, sendo 3199 imagens com o rótulo “chihuahua” e as 2718 imagens com o rótulo “muffin”, disponibilizadas em diversas dimensões, 3 canais e padronizadas no formato “.jpg”. Deste modo, necessitam pré-processadas para ter as dimensões do padrão ImageNet.

Adicionalmente, conforme ilustrado na Figura 50, esse banco de dados é composto por uma grande quantidade de imagens que não são *muffins* e imagens que não são *chihuahuas* e, portanto, apresenta um segundo grande desafio: obriga o modelo entender as diferentes características dentro de cada classe (Cortinhas; Kaggle, 2022).

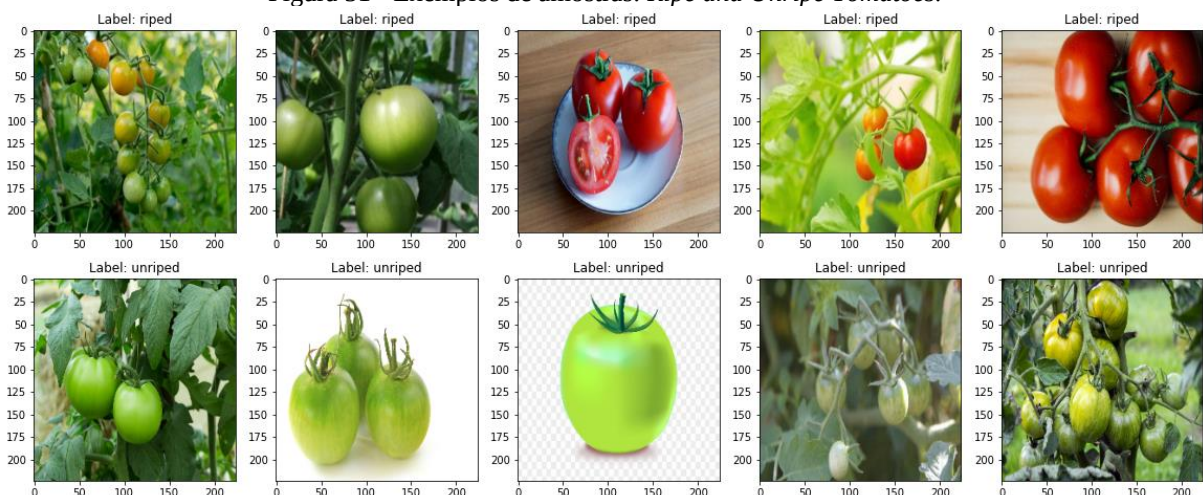
Figura 50 - Exemplos de amostras: *Muffin vs chihuahua*.

Fonte: Elaboração do próprio autor.

6.3.4 Ripe and Unripe Tomatoes

Para classificação de tomates maduros ou não maduros, este banco de dados apresenta 177 imagens, sendo 97 imagens com o rótulo “*ripen*” e “80” com o rótulo “*unripen*”. As imagens também são apresentadas inicialmente com diversas dimensões, 3 canais e todas no formato “.jpeg”. São pré-processadas para atender a dimensões padrão Imagenet.

Muitas imagens deste banco de dados podem possuir ou não informações de fundo e outras possuem rótulos ambíguos ao olho humano, conforme ilustrado na Figura 51 (Suman Kunwar; Kaggle, 2024).

Figura 51 - Exemplos de amostras: *Ripe and Unripe Tomatoes*.

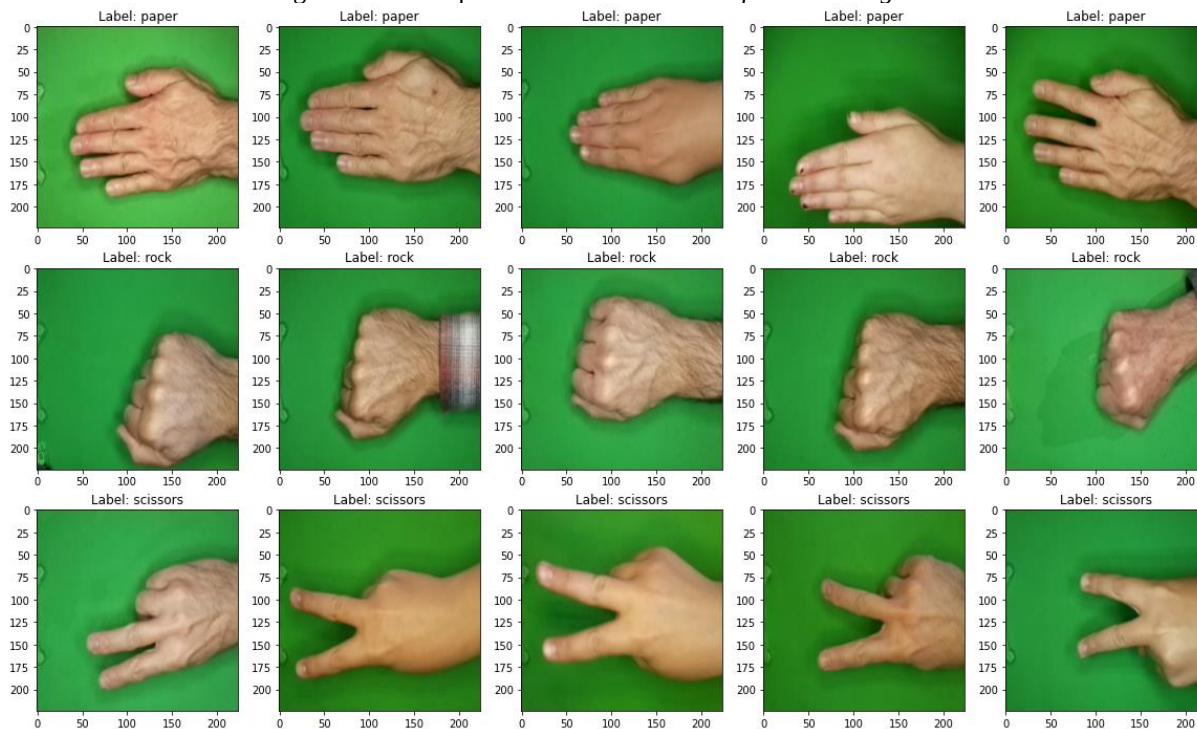
Fonte: Elaboração do próprio autor.

6.3.5 Rock-Paper-Scissors game

Este conjunto de dados contém imagens de gestos manuais do jogo Pedra-Papel-Tesoura. O conjunto de dados contém um total de 2188 imagens, correspondentes aos gestos de mão “*Rock*” (726 imagens), “*Paper*” (710 imagens) e “*Scissors*” (752 imagens).

Conforme ilustrado na Figura 52, todas as imagens possuem fundo verde, iluminação e balanço de branco relativamente consistentes. Além disto, este banco de dados, originalmente, disponibiliza imagens no formato “.png”, 3 canais e em dimensões diversas (de la Bruère-Terreault; Kaggle, 2022).

Figura 52 - Exemplos de amostras: *Rock-Paper-Scissors game*.



Fonte: Elaboração do próprio autor.

6.3.6 Traffic Sign

Este conjunto de dados é utilizado para classificar as diferentes classes de sinalização de trânsito. Composto por 58 classes na totalidade, este banco de dados apresenta uma grande variação do número de amostras por classe, *i.e.*, a classe com maior número de amostras possui 446 imagens enquanto a classe com menos amostras apenas 2 imagens. Ao todo, 4170 imagens foram utilizadas para as etapas de treinamento e teste. Originalmente, as imagens possuem diversos tamanhos, 3 canais e são padronizadas no formato “.png”. A Figura 53 ilustra alguns exemplos do banco de dados (Hemateja; Kaggle, 2021).

Figura 53 - Exemplos de amostras: *Traffic Sign*.

Fonte: Elaboração do próprio autor.

6.3.7 Considerações sobre os resultados: outros bancos de dados

Os bancos supracitados apresentam aos modelos testados diferentes propostas com características particulares; seja a variedade da quantidade de amostras disponibilizada por cada conjunto de treinamento, seja pelo tamanho das classes possíveis e ainda, pela qualidade das imagens, *i.e.*, conjuntos com muitas imagens ruidosas e conjuntos com imagens tratadas.

Na Tabela 10 é apresentado o resumo de desempenhos dos algoritmos VGG ARTMAP *Fuzzy* com *pooling* médio global e da CNN baseada em um modelo clássico da literatura VGG-16.

Em relação ao tempo de treinamento, o modelo profundo baseado na VGG-16, novamente, apresenta um desempenho muito superior, visto que, trata-se de um algoritmo otimizado. Todavia, a acurácia do modelo proposto se mostrou competitivo nos testes. Inclusivamente, o modelo proposto apresentou desempenhos superiores para aplicações com conjunto de dados menores.

Tabela 10 - Acurácias e tempo de treinamento de VGG ARTMAP *Fuzzy* com *pooling* médio global e VGG-16 adaptada.

Banco de dados	Conjunto de treinamento (imagens)	VGG ARTMAP <i>Fuzzy</i>		VGG-16 adaptada	
		Tempo de treinamento (horas:minutos:segundos)	Acurácia	Tempo de treinamento (horas:minutos:segundos)	Acurácia
Brain MRI Images for Brain Tumor Detection	202	0:00:25	90,19%	0:00:07	86,27%
Car vs Bike Classification	3200	00:18:47	97,75%	0:09:43	99%
Muffin vs chihuahua	4733	02:00:10	94,59%	0:02:26	99,49%
Ripe and Unripe Tomatoes	141	0:00:14	97,22%	0:00:05	94,44%
Rock-Paper-Scissors game	1750	0:05:06	90,18%	0:02:04	99,54%
Traffic Sign	3336	1:32:44	93,28%	0:12:41	95,92%

Fonte: Elaboração do próprio autor.

7 CONSIDERAÇÕES FINAIS

7.1 CONSIDERAÇÕES GERAIS

Este trabalho teve por objetivo o desenvolvimento de um modelo de rede neural híbrido para a tarefa de classificação de imagens, denominado de VGG ARTMAP *Fuzzy*. Por se tratar de uma proposta inicial, optou-se pela combinação do aprendizado profundo com a Teoria da Ressonância Adaptativa a partir de modelos clássicos da literatura e não necessariamente os mais otimizados atualmente.

Por intermédio da transferência de aprendizado, este modelo aproveita o vasto conhecimento de uma rede convolucional VGG-16 pré-treinada para realizar a extração de características das informações contidas em imagens que são utilizadas para a classificação, realizada por uma rede ARTMAP *Fuzzy*.

Algoritmos baseados em aprendizado profundo, como as redes convolucionais, se tornaram populares em aplicações de visão computacional, a partir do avanço dos computadores, na disponibilidade dos *big datas* e por apresentarem grande precisão. Entretanto, como na maioria dos casos estes algoritmos são treinados por retropropagação de erro, podem apresentar esquecimento catastrófico, são suscetíveis a ataques adversários e, portanto, possuem confiabilidade questionáveis (Grossberg, 2020).

As técnicas de transferência de aprendizado possibilitam a combinação de redes baseadas na ART com redes profundas convolucionais e aproveitam os benefícios das duas grandes áreas: a sensibilidade para extração de características em uma imagem a partir de uma CNN e uma classificação flexível e estável por uma rede neural ART.

No modelo proposto, a transferência de aprendizado por representação de característica foi utilizada e, portanto, os pesos pré-treinados da VGG-16 para o banco de dados da ImageNet foram utilizados sem ajustes. Além disto, a teoria difusa faz-se necessária na rede ARTMAP para classificação de dados analógicos.

A arquitetura da VGG-ARTMAP *Fuzzy* é composta por 5 blocos convolucionais provenientes da VGG-16, além de dois módulos ART e um módulo inter-ART, da ARTMAP *Fuzzy*. Entre a saída da VGG-16 e a entrada da ARTMAP *Fuzzy* foram utilizadas duas técnicas que proporcionaram variantes do modelo: a VGG-ARTMAP *Fuzzy* com achatamento e a VGG-ARTMAP *Fuzzy* com *pooling* médio global.

Para validação das propostas, foram realizadas simulações em diferentes bancos de dados de visão computacional com a finalidade de explorar aplicações em diversos contextos.

Como em visão computacional o tempo de treinamento é um assunto rotineiro, além da acurácia, os tempos de treinamento e testes foram considerados.

Como apresentados na Seção 6, alguns bancos de dados possuem diversas ambiguidades em imagens que, confusas até mesmo ao olho humano, comprometeram o desempenho do aprendizado da máquina. Entretanto, os resultados exibem uma acurácia acima de 90% em muitas das simulações e aponta que o modelo VGG ARTMAP *Fuzzy* apresentou bastante robustez em detectar diferentes características das imagens em diferentes ocasiões e, conseqüentemente, possibilitou um bom desempenho na tarefa de classificação de imagens.

No geral, o modelo VGG ARTMAP *Fuzzy* demonstrou ser capaz de classificar imagens com precisão competitiva às outras técnicas de classificação puramente de aprendizado profundo.

7.2 VARIANTES

A variante VGG-ARTMAP *Fuzzy* com achatamento demonstrou ter maior acurácia em relação ao modelo com *pooling* médio global. Esse efeito era esperado, uma vez que, em cada mapa de características de dimensão 7×7 disponibilizado pela VGG-16, é considerado a relação de vizinhança entre os pixels que, por outro lado, são sintetizados a um único valor ao utilizar o *pooling* médio global.

Em outros termos, o *pooling* médio global sintetiza “o quanto” de cada característica - entre as 512 disponibilizadas pela VGG-16, está presente na imagem de entrada. Por outro lado, ao remanejar diretamente para um vetor resultante por operação de achatamento, além “do quanto” é possível distinguir o “como” essas características estão disponibilizadas. Em contrapartida, considerar toda essa informação de vizinhança tem um custo computacional muito mais caro, uma vez que, o vetor de saída, disponibilizado pela VGG-16, ao realizar o achatamento é 49 vezes maior do que o da variante que o sintetiza por intermédio do *pooling* médio global.

Ao considerar a operação de achatamento, apesar do ganho na precisão ser relevante, há uma perda considerável de desempenho; não só em relação ao tempo de treinamento, mas também ao tempo de resposta da rede para o prognóstico de uma entrada. Deste modo, aplicações que o tempo de resposta precisa ser curto, como classificação de placas de trânsito em veículos autônomos por exemplo, é mais recomendado a escolha pela utilização da variante composta por *pooling* médio global. Por outro lado, em aplicações em que alguns segundos a mais para ter o prognóstico - como uma análise de fratura em um raio-X, por

exemplo - não seja comprometedor, é válido utilizar uma variante que terá uma acurácia maior, mesmo que necessite de mais tempo de processamento.

7.3 PROCESSAMENTO COMPUTACIONAL

A plataforma *TensorFlow* permite a aplicação da rede convolucional VGG de forma otimizada; trata-se de um algoritmo com anos de aperfeiçoamento que, em conjunto com a GPU NVIDIA GeForce RTX 3070, opera em programação paralela (*multithread*) e, portanto, utiliza todo o recurso computacional disponível para o processamento dos dados. Por outro lado, a etapa do código composta pela ARTMAP *Fuzzy*, opera de forma não otimizada, *i.e.*, programação sequencial (*single thread*). Deste modo, os desempenhos, em relação a tempo de treinamento e teste, são muito dispersos ao comparar uma VGG pura com uma VGG adicionada com uma ARTMAP *Fuzzy*.

Além disso, redes da família ART são algoritmos incrementais. Isto posto, foi observado que o aprendizado entre uma amostra e outra possui maior rapidez no início do treinamento, se tornando mais lento à medida que prossegue, visto que a rede se expande quando novos neurônios (ou categorias) são ativados. Este comportamento é esperado, entretanto, para grandes bancos de dados, comuns na área de visão computacional, esta característica deve ser minuciosamente considerada.

Os parâmetros de vigilância da rede ARTMAP *Fuzzy*, alteram a sensibilidade da rede, de modo a ser mais ou menos generalista, o que influencia diretamente no número de categorias criadas e, conseqüentemente, no custo computacional exigido. Entretanto, nas diferentes simulações foram utilizados praticamente os mesmos valores pseudo-ótimos para estes parâmetros, com pouca ou nenhuma alteração, de modo que as comparações entre as variantes propostas fossem mais justas.

7.4 ATAQUES ADVERSÁRIOS

Os resultados das simulações relacionadas à ataques adversários apresentados na Seção 6, demonstram que tanto o algoritmo da rede neural profunda VGG pura quanto o modelo híbrido possuem vulnerabilidade a estes ataques. Os modelos não classificam corretamente imagens com ruídos adversários. Mesmo que o conteúdo da imagem ruidosa seja de fácil classificação ao olho humano, a inteligência artificial perde a capacidade de reconhecer corretamente as características presentes nos padrões de entrada, proporcionando uma classificação equivocada na maioria dos testes.

Em relação ao modelo híbrido proposto, esta vulnerabilidade também já era esperada, visto que, a etapa de reconhecimento de características é desempenhada por uma rede neural

convolucional e, conseqüentemente, como apontado na literatura, é susceptível a estes ataques. Conseqüentemente, o classificador não será capaz de realizar uma predição correta visto que o extrator de características fornece em sua saída um padrão com valores equivocados.

Uma das saídas para essa questão, como sugerido em Bai, *et al.* (2021), é o treinamento adversário. Segundo os autores esta é uma das abordagens mais eficazes para modelos de aprendizado profundo se defender contra os ataques adversários. Modelos treinados adversamente têm melhores desempenhos quando enfrentam exemplos adversários ao comparados com modelos treinados de modo padrão.

No entanto, redes profundas clássicas utilizam o ajuste de pesos de forma não local, logo, a implementação de estratégias de defesa baseadas em treinamento futuro pode levar à degradação do desempenho e eficiência do modelo em questão.

Neste aspecto as redes da família ART possuem grande vantagem. Devido aos seus sistemas de atenção e orientação, as memórias aprendidas são dinamicamente autoestabilizadas e o ajuste dos pesos são realizados localmente. Esta característica permite treinamentos futuros para adicionar robustez a ataques adversários sem comprometer o conhecimento adquirido anteriormente.

7.5 SUGESTÕES PARA TRABALHOS FUTUROS

Nesta pesquisa, procurou-se estudar e compreender o comportamento das técnicas do aprendizado de máquina para resolver o problema-alvo (classificação de imagens), mais voltado para as questões sobre a viabilidade no âmbito da resolubilidade, ou seja, quais são o alcance e a possibilidade da aplicação desta abordagem. Entendendo-se como suprida esta viabilidade, considera-se não menos importante a rapidez. Esta etapa completa um ciclo ideal e torna-se, por certo, plausível mediante a inclusão de várias melhorias assim destacadas que compreendem as sugestões para trabalhos futuros nas duas perspectivas analisadas: o tempo de processamento e a acurácia.

Em relação ao tempo computacional exigido, a estratégia de utilização de uma camada obtida pelo *pooling* médio global ao invés de achatamento, demonstra que reduzir o tamanho do vetor de entrada, apresentado para a ARTMAP *Fuzzy*, permite maior velocidade de processamento entre as amostras.

A ARTMAP *Fuzzy* é uma das redes da família ART que necessita do código de complemento na etapa de pré-processamento. Isso significa que um vetor de tamanho 512 da saída da VGG terá tamanho 1024 na entrada da ARTMAP *Fuzzy*, do mesmo modo que, um

vetor de saída de tamanho 25088 se torna um vetor de entrada no tamanho 50176 - no caso da variante que se utiliza a operação de achatamento. Neste aspecto, redes neurais da família ART que não exigem código complemento, como por exemplo a ARTMAP Gaussiana e a ARTMAP Bayesiana, podem proporcionar melhores tempos de respostas, tanto de treinamento quanto as de predições pós treinamento.

As simulações demonstram que a opção de se utilizar a operação de achatamento ao invés de camada de *pooling* médio global criou-se um dilema: melhor tempo de resposta *versus* maior precisão.

Uma alternativa intermediária pode ser adicionar mais uma camada convolucional na saída do extrator de características para que seja disponibilizado mapas de características menores, de tamanho 3×3 por exemplo. Neste caso, a relação de vizinhança não é perdida e cada mapa de características fornece 9 valores ao invés de 49, como é o caso da VGG-16. Outra opção é simplesmente diminuir o número de mapas de características liberados para o classificador. Entretanto, adicionar mais uma camada convolucional, é equivalente adicionar mais parâmetros treináveis e, portanto, essa escolha deve ser minuciosamente avaliada.

Outra estratégia que pode ser colocada em prática é a otimização do algoritmo para extração máxima do poder de *hardware*. Como supracitado, apesar da etapa de extração de características, promovida pela VGG-16 estar bem otimizada, a etapa de classificação promovida pela ARTMAP Fuzzy possui um grande potencial de melhoria.

Em relação à acurácia do algoritmo, a técnica de validação de respostas de forma cruzada pode ser uma estratégia que contribua em uma melhoria geral da acurácia do algoritmo. Entretanto, exigiria maior tempo de processamento, uma vez que, necessite de treinamento de múltiplos classificadores.

Como os parâmetros de treinamento do classificador ARTMAP Fuzzy também não foram essencialmente explorados, um estudo futuro pode ser necessário para investigar o quão genérico ou privativo um classificador possa ser, quando aplicados a treinamentos de bancos de dados maiores ou menores. Este comportamento é ajustado de forma simples por adequações dos valores dos parâmetros de vigilâncias.

Em relação ao ajuste “ótimo” dos principais parâmetros da ARTMAP-Fuzzy, a principal sintonia refere-se à adoção do parâmetro de vigilância do módulo ARTa-Fuzzy “ótimo” retirando-se o dispositivo incremental ε . Esta execução costuma-se ser muito demorada sem, a rigor, oferecer resultados mais qualitativos. Esta modalidade tem sido proposta na literatura especializada visando à resolução de problemas com características particulares que passou a ser usada com grande frequência sem uma razão tangível. Também,

“criar” neurônios na ANN ARTa-Fuzzy, somente quando for necessário rigorosamente, ou seja, inibir a “criação” de um novo neurônio quando não for vantajoso. Neste caso, deve-se desenvolver uma métrica adequada para esta incumbência.

Ressalta-se que a qualidade das soluções é altamente dependente da forma como as informações de entrada e de saída são organizadas e pré-processadas. Pode-se, portanto, desenvolver melhor forma de proceder na elaboração da base de dados. Talvez seja esta etapa a mais importante de melhoria dos resultados. As ANN da família ART podem ser interpretadas como um *naïve* paradoxo, *viz.*, é muito fácil de ser compreendido o seu funcionamento, mas apresenta alta complexidade na execução da sintonia dos parâmetros da ANN.

Deve-se destacar que são arquiteturas neurais altamente eficientes em termos de precisão, estabilidade, plasticidade, celeridade que, decerto, não encontra, na atualidade, concorrência relevante. Repetindo, podem-se obter resultados altamente competitivos, obviamente, recorrendo à inclusão de inovações assertivas. A autoria destas ANN (Grossberg, 2020) tem enfaticamente reportado sobre a inspiração desta ANN que é advinda de uma analogia funcional de parte do cérebro. Como este assunto ainda não se esgotou, tem-se à disposição um universo aberto para a inovação. O Prof. Grossberg é um cientista cognitivo, neurocientista, matemático, engenheiro biomédico e tecnólogo neuromórfico. Trata-se, portanto, de ótimas credenciais.

Sugere-se usar imagens em alto contraste (em cor preta) dispensando os tons-cinza. Estas imagens podem ser geradas a partir do uso de filtros considerando as cores aditivas RGB (ciano, vermelho, verde) e, se necessário, pode ser adicionada uma imagem contendo os detalhes. Estas imagens filtradas são composições que guardam grandes diferenciais entre si o que facilita a classificação das imagens com melhor qualidade.

Sobre a inclusão de dispositivo de pós-processamento, sugere-se, por exemplo, a implementação baseada na teoria de evidência de Dempster-Shafer (Barnett, 2008), constituindo-se num refinamento da solução final com vista a aumentar a precisão das respostas. Deve-se tentar, igualmente, o uso das transformadas *wavelets*.

Complementarmente, o modelo - ou o conceito - pode ser expandido para outros campos de visão computacional, *e.g.*: detecção de objetos por caixa delimitadora (*Bounding Box*) em imagens ou até mesmo em vídeos; além de classificação em outros meios, como sonoros e textuais.

REFERÊNCIAS

- ABADI, Martín *et al.* Tensorflow: Large-scale machine learning on heterogeneous distributed systems. **arXiv preprint arXiv:1603.04467**, [s. l.], 2016.
- ABDEL-HAMID, Ossama *et al.* Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. *In:* , 2012. **2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)**. [S. l.]: IEEE, 2012. p. 4277–4280.
- AKHTAR, N; MIAN, A. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. **IEEE Access**, [s. l.], v. 6, p. 14410–14430, 2018.
- ALOM, Md Zahangir *et al.* A state-of-the-art survey on deep learning theory and architectures. **Electronics**, [s. l.], v. 8, n. 3, p. 292, 2019.
- ALURU V N M, Hemateja; KAGGLE. **Traffic Sign Dataset - Classification**. [S. l.], 2021.
- ARAUJO, André; NORRIS, Wade; SIM, Jack. Computing receptive fields of convolutional neural networks. **Distill**, [s. l.], v. 4, n. 11, p. e21, 2019.
- BACKES, André Ricardo; SÁ JUNIOR, Jarbas Joaci de Mesquita. **Introdução à visão computacional usando Matlab**. Rio de Janeiro: Alta Books Editora, 2019.
- BAI, Tao *et al.* Recent advances in adversarial training for adversarial robustness. **arXiv preprint arXiv:2102.01356**, [s. l.], 2021.
- BARNETT, Jeffrey A. Computational Methods for A Mathematical Theory of Evidence. *In:* YAGER ROLAND R. AND LIU, Liping (org.). **Classic Works of the Dempster-Shafer Theory of Belief Functions**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 197–216. Disponível em: https://doi.org/10.1007/978-3-540-44792-4_8.
- BU, Z *et al.* Encrypted Network Traffic Classification Using Deep and Parallel Network-in-Network Models. **IEEE Access**, [s. l.], v. 8, p. 132950–132959, 2020.
- CARPENTER, G A *et al.* Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. **IEEE Transactions on Neural Networks**, [s. l.], v. 3, n. 5, p. 698–713, 1992.
- CARPENTER, Gail A; GROSSBERG, Stephen. A massively parallel architecture for a self-organizing neural pattern recognition machine. **Computer vision, graphics, and image processing**, [s. l.], v. 37, n. 1, p. 54–115, 1987a.
- CARPENTER, Gail A; GROSSBERG, Stephen. ART 2: self-organization of stable category recognition codes for analog input patterns. **Applied Optics**, [s. l.], v. 26, n. 23, p. 4919–4930, 1987b. Disponível em: <http://ao.osa.org/abstract.cfm?URI=ao-26-23-4919>.
- CARPENTER, Gail A; GROSSBERG, Stephen; REYNOLDS, John H. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. **Neural Networks**, [s. l.], v. 4, n. 5, p. 565–588, 1991. Disponível em: <http://www.sciencedirect.com/science/article/pii/089360809190012T>.

CARPENTER, Gail A; GROSSBERG, Stephen; ROSEN, David B. ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition. **Neural Networks**, [s. l.], v. 4, n. 4, p. 493–504, 1991a.

CARPENTER, Gail A; GROSSBERG, Stephen; ROSEN, David B. Fuzzy ART: An adaptive resonance algorithm for rapid, stable classification of analog patterns. *In:* , 1991b. **International Joint Conference on Neural Networks**. [S. l.: s. n.], 1991. p. 411–416.

CARPENTER, Gail A; GROSSBERG, Stephen; ROSEN, David B. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. **Neural Networks**, [s. l.], v. 4, n. 6, p. 759–771, 1991c. Disponível em: <http://www.sciencedirect.com/science/article/pii/089360809190056B>.

CHAKRABORTY, Anirban *et al.* A survey on adversarial attacks and defenses. **CAAI Transactions on Intelligence Technology**, [s. l.], v. 6, n. 1, p. 25–45, 2021.

CHEN, M *et al.* Disease Prediction by Machine Learning Over Big Data From Healthcare Communities. **IEEE Access**, [s. l.], v. 5, p. 8869–8879, 2017.

CORTINHAS, Samuel; KAGGLE. **Muffin vs chihuahua**. [S. l.], 2022. Disponível em: <https://www.kaggle.com/datasets/samueltcortinhas/muffin-vs-chihuahua-image-classification>. Acesso em: 8 fev. 2024.

COSTA, Joana C *et al.* How Deep Learning Sees the World: A Survey on Adversarial Attacks & Defenses. **IEEE Access**, [s. l.], v. 12, p. 61113–61136, 2024.

DA SILVA, Leonardo Enzo Brito; ELNABARAWY, Islam; WUNSCH, Donald C. A Survey of Adaptive Resonance Theory Neural Network Models for Engineering Applications. [s. l.], 2019. Disponível em: <http://arxiv.org/abs/1905.11437>.

DE LA BRUÈRE-TERREAULT, Julien; KAGGLE. **Rock-Paper-Scissors Images**. [S. l.], 2022.

DECANINI, J G M S; TONELLI-NETO, M s.; MINUSSI, C R. Robust fault diagnosis in power distribution systems based on fuzzy ARTMAP neural network-aided evidence theory. **IET Generation, Transmission & Distribution**, [s. l.], v. 6, n. 11, p. 1112–1120, 2012.

DEEPNETS; KAGGLE. **Car vs Bike Classification Dataset**. [S. l.], 2022.

DENG, Jia *et al.* Imagenet: A large-scale hierarchical image database. *In:* , 2009. **2009 IEEE conference on computer vision and pattern recognition**. [S. l.]: Ieee, 2009. p. 248–255.

DENKER, John *et al.* Neural network recognizer for handwritten zip code digits. **Advances in neural information processing systems**, [s. l.], v. 1, 1988.

DONG, Shi; WANG, Ping; ABBAS, Khushnood. A survey on deep learning and its applications. **Computer Science Review**, [s. l.], v. 40, p. 100379, 2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574013721000198>.

DUBEY, Arun Kumar; JAIN, Vanita. Comparative Study of Convolution Neural Network's Relu and Leaky-Relu Activation Functions. *In: , 2019, Singapore. (Sukumar Mishra, Yog Raj Sood, & Anuradha Tomar, Org.)* **Applications of Computing, Automation and Wireless Systems in Electrical Engineering**. Singapore: Springer Singapore, 2019. p. 873–880.

EYKHOLT, Kevin *et al.* Robust Physical-World Attacks on Deep Learning Visual Classification. *In: , 2018. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S. l.: s. n.], 2018. p. 1625–1634.

FUKUSHIMA, Kunihiko; MIYAKE, Sei. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. *In: COMPETITION AND COOPERATION IN NEURAL NETS*. [S. l.]: Springer, 1982. p. 267–285.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep learning**. [S. l.]: MIT press, 2016.

GOODFELLOW, Ian J; SHLENS, Jonathon; SZEGEDY, Christian. Explaining and harnessing adversarial examples. **arXiv preprint arXiv:1412.6572**, [s. l.], 2014.

GOPALAKRISHNAN, Kasthurirangan *et al.* Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. **Construction and Building Materials**, [s. l.], v. 157, p. 322–330, 2017. Disponível em: Acesso em: 20 dez. 2022.

GROSSBERG, Stephen. A Path Toward Explainable AI and Autonomous Adaptive Intelligence: Deep Learning, Adaptive Resonance, and Models of Perception, Emotion, and Action. **Frontiers in Neurorobotics**, [s. l.], v. 14, p. 36, 2020.

GROSSBERG, S. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. **Biological Cybernetics**, [s. l.], v. 23, n. 3, p. 121–134, 1976a.

GROSSBERG, S. Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions. **Biological Cybernetics**, [s. l.], v. 23, n. 4, p. 187–202, 1976b.

GUPTA, Riyanshi; BHARDWAJ, Kartik Krishna; SHARMA, Deepak Kumar. Transfer Learning. **Machine Learning and Big Data: Concepts, Algorithms, Tools and Applications**, [s. l.], p. 335–360, 2020.

HE, Kaiming *et al.* Deep residual learning for image recognition. *In: , 2016. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S. l.: s. n.], 2016. p. 770–778.

HEBB, Donald Olding. **The organization of behavior: a neuropsychological theory**. Psychology: J. Wiley; Chapman & Hall, 1949.

HINTON, Geoffrey *et al.* Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. **IEEE Signal processing magazine**, [s. l.], v. 29, n. 6, p. 82–97, 2012.

HINTON, Geoffrey E *et al.* Improving neural networks by preventing co-adaptation of feature detectors. **arXiv preprint arXiv:1207.0580**, [s. l.], 2012.

HINTON, Geoffrey; VINYALS, Oriol; DEAN, Jeff. Distilling the knowledge in a neural network. **arXiv preprint arXiv:1503.02531**, [s. l.], v. 2, n. 7, 2015.

HUANG, Gao *et al.* Densely connected convolutional networks. *In:* , 2017. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S. l.: s. n.], 2017. p. 4700–4708.

HUANG, K -Y. *et al.* Speech Emotion Recognition Using Deep Neural Network Considering Verbal and Nonverbal Speech Sounds. *In:* , 2019. **ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S. l.: s. n.], 2019. p. 5866–5870.

HUANG, Shih-Chia; LE, Trung-Hieu. **Principles and Labs for Deep Learning**. [S. l.]: Academic Press, 2021.

JORDAN, M I. Serial Ordar: A Parallel Distributed Processing Approach. **Tech. Rep. 8604**, [s. l.], 1986.

KALCHBRENNER, Nal; GREFFENSTETTE, Edward; BLUNSOM, Phil. A convolutional neural network for modelling sentences. **arXiv preprint arXiv:1404.2188**, [s. l.], 2014.

KHAN, Salman *et al.* A guide to convolutional neural networks for computer vision. **Synthesis lectures on computer vision**, [s. l.], v. 8, n. 1, p. 1–207, 2018.

KINGMA, Diederik P; BA, Jimmy. Adam: A Method for Stochastic Optimization. **CoRR**, [s. l.], v. abs/1412.6980, 2014. Disponível em: <https://api.semanticscholar.org/CorpusID:6628106>.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, [s. l.], v. 25, 2012.

KURAKIN, Alexey; GOODFELLOW, Ian; BENGIO, Samy. Adversarial examples in the physical world. [s. l.], 2016a.

KURAKIN, Alexey; GOODFELLOW, Ian J; BENGIO, Samy. Adversarial examples in the physical world. *In:* ARTIFICIAL INTELLIGENCE SAFETY AND SECURITY. [S. l.]: Chapman and Hall/CRC, 2018. p. 99–112.

KURAKIN, Alexey; GOODFELLOW, Ian; BENGIO, Samy. Adversarial machine learning at scale. **arXiv preprint arXiv:1611.01236**, [s. l.], 2016b.

LE, Hung; BORJI, Ali. What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks?. **arXiv preprint arXiv:1705.07049**, [s. l.], 2017.

LECUN, Yann *et al.* Backpropagation applied to handwritten zip code recognition. **Neural computation**, [s. l.], v. 1, n. 4, p. 541–551, 1989.

LECUN, Yann. Generalization and network design strategies. **Connectionism in perspective**, [s. l.], v. 19, n. 143–155, p. 18, 1989.

LECUN, Yann *et al.* Gradient-Based Learning Applied to Document Recognition. [s. l.], 1998. Disponível em: Acesso em: 17 jul. 2022.

LECUN, Yann; CORTES, Corinna; BURGESS, C J. MNIST handwritten digit database. **ATT Labs [Online]**. Available: <http://yann.lecun.com/exdb/mnist>, [s. l.], v. 2, 2010.

LI, Z *et al.* A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. **IEEE Transactions on Neural Networks and Learning Systems**, [s. l.], v. 33, n. 12, p. 6999–7019, 2022.

LIN, Min; CHEN, Qiang; YAN, Shuicheng. Network in network. **arXiv preprint arXiv:1312.4400**, [s. l.], 2013.

MADRY, Aleksander *et al.* Towards deep learning models resistant to adversarial attacks. **arXiv preprint arXiv:1706.06083**, [s. l.], 2017.

MCCLOSKEY, Michael; COHEN, Neal J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. **Psychology of Learning and Motivation - Advances in Research and Theory**, [s. l.], v. 24, n. C, p. 109–165, 1989. Disponível em: Acesso em: 14 fev. 2022.

MCCULLOCH, Warren S; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, [s. l.], v. 5, n. 4, p. 115–133, 1943. Disponível em: <https://doi.org/10.1007/BF02478259>.

MEHROTRA, R; NAMUDURI, K R; RANGANATHAN, N. Gabor filter-based edge detection. **Pattern Recognition**, [s. l.], v. 25, n. 12, p. 1479–1494, 1992. Disponível em: <https://www.sciencedirect.com/science/article/pii/003132039290121X>.

MOORE, Barbara. ART 1 and pattern clustering. In: , 1989. **Proceedings of the 1988 connectionist models summer school**. [S. l.]: Morgan Kaufmann Publishers San Mateo, CA, 1989. p. 174–185.

NAIR, Vinod; HINTON, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In: , 2010. **Icml**. [S. l.: s. n.], 2010.

NAVONEEL CHAKRABARTY; KAGGLE. **Brain MRI Images for Brain Tumor Detection**. [S. l.], 2019.

O'SHEA, Keiron; NASH, Ryan. An introduction to convolutional neural networks. **arXiv preprint arXiv:1511.08458**, [s. l.], 2015.

PAN, S J; YANG, Q. A Survey on Transfer Learning. **IEEE Transactions on Knowledge and Data Engineering**, [s. l.], v. 22, n. 10, p. 1345–1359, 2010.

PAPERNOT, Nicolas *et al.* Distillation as a defense to adversarial perturbations against deep neural networks. *In:* , 2016a. **2016 IEEE Symposium on security and privacy (SP)**. [S. l.]: IEEE, 2016. p. 582–597.

PAPERNOT, Nicolas *et al.* The Limitations of Deep Learning in Adversarial Settings. *In:* , 2016b. **2016 IEEE European Symposium on Security and Privacy (EuroS&P)**. [S. l.: s. n.], 2016. p. 372–387.

PORIA, Soujanya *et al.* A deeper look into sarcastic tweets using deep convolutional neural networks. **arXiv preprint arXiv:1610.08815**, [s. l.], 2016.

RIBANI, Ricardo; MARENGONI, Mauricio. A survey of transfer learning for convolutional neural networks. *In:* , 2019. **2019 32nd SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)**. [S. l.]: IEEE, 2019. p. 47–57.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, [s. l.], v. 65, n. 6, p. 386, 1958.

RUSSAKOVSKY, Olga *et al.* ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision**, [s. l.], v. 115, n. 3, p. 211–252, 2015a. Disponível em: <https://doi.org/10.1007/s11263-015-0816-y>.

RUSSAKOVSKY, Olga *et al.* ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision**, [s. l.], v. 115, p. 211–252, 2015b. Disponível em: <http://image-net.org/challenges/LSVRC/>.

SAMANGOUEI, Pouya; KABKAB, Maya; CHELLAPPA, Rama. Defense-gan: Protecting classifiers against adversarial attacks using generative models. **arXiv preprint arXiv:1805.06605**, [s. l.], 2018.

SERRANO-GOTARREDONA, Teresa; LINARES-BARRANCO, Bernabé. A Modified ART 1 Algorithm more Suitable for VLSI Implementations. **Neural Networks**, [s. l.], v. 9, n. 6, p. 1025–1043, 1996.

SIAU, Keng; WANG, Weiyu. Building trust in artificial intelligence, machine learning, and robotics. **Cutter business technology journal**, [s. l.], v. 31, n. 2, p. 47–53, 2018.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, [s. l.], 2014.

SRIVASTAVA, Nitish *et al.* Dropout: a simple way to prevent neural networks from overfitting. **The Journal of machine learning research**, [s. l.], v. 15, n. 1, p. 1929–1958, 2014.

SRIVASTAVA, Rupesh Kumar; GREFF, Klaus; SCHMIDHUBER, Jürgen. Highway networks. **arXiv preprint arXiv:1505.00387**, [s. l.], 2015.

SU, Jiawei; VARGAS, Danilo Vasconcellos; SAKURAI, Kouichi. One pixel attack for fooling deep neural networks. **IEEE Transactions on Evolutionary Computation**, [s. l.], v. 23, n. 5, p. 828–841, 2019.

SUMAN KUNWAR; KAGGLE. **Ripe and Unripe Tomatoes Dataset**. [S. l.], 2024.

SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to sequence learning with neural networks. **Advances in neural information processing systems**, [s. l.], v. 27, 2014.

SZEGEDY, Christian *et al.* Going deeper with convolutions. *In:* , 2015. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S. l.: s. n.], 2015. p. 1–9.

SZEGEDY, Christian *et al.* Intriguing properties of neural networks. **arXiv preprint arXiv:1312.6199**, [s. l.], 2013.

TAMMINA, Srikanth. Transfer learning using vgg-16 with deep convolutional neural network for classifying images. **International Journal of Scientific and Research Publications (IJSRP)**, [s. l.], v. 9, n. 10, p. 143–150, 2019.

THEPADE, Sudeep D *et al.* Face presentation attack identification optimization with adjusting convolution blocks in VGG networks. **Intelligent Systems with Applications**, [s. l.], v. 16, p. 200107, 2022. Disponível em: <https://www.sciencedirect.com/science/article/pii/S266730532200045X>.

WANG, Peng *et al.* Semantic Clustering and Convolutional Neural Network for Short Text Categorization. *In:* , 2015, Beijing, China. **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)**. Beijing, China: Association for Computational Linguistics, 2015. p. 352–357. Disponível em: <https://aclanthology.org/P15-2058>.

WEISS, Karl; KHOSHGOFTAAR, Taghi M; WANG, DingDing. A survey of transfer learning. **Journal of Big data**, [s. l.], v. 3, n. 1, p. 1–40, 2016.

WERBOS, Paul. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. **Ph. D. dissertation, Harvard University**, [s. l.], 1974.

XIE, Saining *et al.* Aggregated residual transformations for deep neural networks. *In:* , 2017. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S. l.: s. n.], 2017. p. 1492–1500.

XIONG, Wayne *et al.* The Microsoft 2017 conversational speech recognition system. *In:* , 2018. **2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)**. [S. l.]: IEEE, 2018. p. 5934–5938.

YANG, Qiaoning *et al.* Deep convolution neural network-based transfer learning method for civil infrastructure crack detection. **Automation in construction**, Amsterdam, v. 116, p. 103199, 2020.

YE, Haijian *et al.* Vegetable Pest Image Recognition Method Based on Improved VGG Convolution Neural Network. **Journal of physics. Conference series**, Bristol, v. 1237, n. 3, p. 32018, 2019.

YOSINSKI, Jason *et al.* How transferable are features in deep neural networks?. **Advances in neural information processing systems**, [s. l.], v. 27, 2014.

ZHANG, Aston *et al.* Dive into deep learning. **arXiv preprint arXiv:2106.11342**, [s. l.], 2021.