UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

SUPERANDO O ESQUECIMENTO CATASTRÓFICO EM REDES NEURAIS CONVOLUCIONAIS PARA O APRENDIZADO INCREMENTAL DE TAREFAS COM ADAPTADORES LOW-RANK

EVERTON LIMA ALEIXO

SUPERANDO O ESQUECIMENTO CATASTRÓFICO EM REDES NEURAIS CONVOLUCIONAIS PARA O APRENDIZADO INCREMENTAL DE TAREFAS COM ADAPTADORES LOW-RANK

Tese apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas, Campus Universitário Senador Arthur Virgílio Filho, como requisito parcial para a obtenção do grau de Doutor em Informática.

ORIENTADOR: JUAN GABRIEL COLONNA

Manaus - AM

Fevereiro de 2025

Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

A366s Aleixo, Everton Lima
Superando o esquecimento catastrófico em redes neurais convolucionais para o aprendizado incremental de tarefas com adaptadores low-rank / Everton Lima Aleixo. - 2025.
155 f. ; 31 cm.

Orientador(a): Juan Gabriel Colonna.
Tese (doutorado) - Universidade Federal do Amazonas, Programa de Pós-Graduação em Informática, Manaus, 2025.

1. Aprendizado Continuo. 2. Esquecimento catastrófico. 3. Low-Rank adapters. 4. Redes Convolucionais. 5. Deep Learning. I. Colonna, Juan Gabriel. II. Universidade Federal do Amazonas. Programa de Pós-Graduação em Informática. III. Título

---

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"SUPERANDO O ESQUECIMENTO CATASTRÓFICO EM REDES NEURAIS CONVOLUCIONAIS PARA O APRENDIZADO INCREMENTAL DE TAREFAS COM ADAPTADORES LOW-RANK"

EVERTON LIMA ALEIXO

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Juan Gabriel Colonna - **Presidente**

Prof. Dr. Eduardo James Pereira Souto - **Membro Interno**

Prof. Dr. Mário Salvatierra Junior - **Membro Externo**

Prof. Dr. Carlos Maurício Serodio Figueiredo - **Membro Externo**

Prof. Dr. Horácio Antônio Braga Fernandes de Oliveira - **Membro Interno**

Manaus, 28 de março de 2025.

Documento assinado eletronicamente por **Carlos Maurício Seródio Figueiredo**, **Usuário Externo**, em 01/04/2025, às 09:19, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8

de outubro de 2015.

Documento assinado eletronicamente por **Mário Salvatierra Júnior**, **Professor do Magistério Superior**, em 01/04/2025, às 09:44, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Horácio Antônio Braga Fernandes de Oliveira**, **Professor do Magistério Superior**, em 01/04/2025, às 14:12, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Eduardo James Pereira Souto**, **Professor do Magistério Superior**, em 02/04/2025, às 08:18, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Juan Gabriel Colonna**, **Professor do Magistério Superior**, em 03/04/2025, às 13:04, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Maria do Perpétuo Socorro Vasconcelos Palheta**, **Secretária em exercício**, em 24/04/2025, às 20:28, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2524938** e o código CRC **D89FAEB0**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Dedico este trabalho à minha amada esposa, Lia Alves Hazan, à minha família – meu pai, José Lima Aleixo, minha mãe, Guaraciara Goreti Lima, e meu irmão, Emerson Lima Aleixo – ao meu orientador e amigo, Prof. Dr. Juan Gabriel Colonna, e a todos os meus amigos e colegas. Em especial, dedico também àqueles que conheci em Manaus e que me apoiaram nesta jornada.

Agradecimentos

A jornada da construção desta tese foi desafiadora e repleta de aprendizados, sempre sustentada pelo apoio indispensável de muitas pessoas, às quais expresso minha profunda gratidão.

Primeiramente, agradeço a Deus, cuja presença e força me sustentaram ao longo desta caminhada, e à minha família, meu alicerce inabalável: meu pai, José Lima Aleixo, minha mãe, Guaraciara Goreti Lima, e meu irmão, Emerson Lima Aleixo. Seu amor, apoio e sacrifícios foram fundamentais para que eu chegasse até aqui. Vocês sempre acreditaram em mim, mesmo quando enfrentei dúvidas e incertezas, e por isso sou eternamente grato. Esta conquista é tão minha quanto de vocês.

À minha amada esposa, Lia Alves Hazan, cuja paciência, carinho e incentivo incondicional me acompanharam em cada etapa desta jornada. Sua presença ao meu lado tornou os desafios mais leves e as conquistas ainda mais significativas. Além disso, não posso deixar de expressar minha gratidão aos seus pais, Edson Jacques Hazan e Eliane da Costa Alves, que sempre foram uma fonte de inspiração, incentivando-me a buscar este objetivo com determinação e coragem.

Ao meu orientador e amigo, Prof. Dr. Juan Gabriel Colonna, por sua orientação dedicada, paciência e incentivo inabaláveis. Seu compromisso com a excelência acadêmica e sua generosidade ao compartilhar conhecimento foram essenciais para a realização deste trabalho, sempre me desafiando a ir além e aprimorar minhas ideias.

Aos meus amigos e colegas de pesquisa, que compartilharam comigo momentos de aprendizado, troca de conhecimento e apoio mútuo. Em especial, sou grato aos que conheci em Manaus, como Everlandio Fernandes e Reinaldo Kavlac, que me acolheram e estiveram ao meu lado nessa jornada, tornando-a mais enriquecedora e especial.

Agradeço também à Universidade Federal do Amazonas (UFAM), que proporcionou o ambiente acadêmico necessário para o desenvolvimento desta pesquisa. Um reconhecimento especial ao SIDIA – Instituto de Ciência e Tecnologia, pelo incentivo e apoio que tornaram possível a execução deste estudo.

Sou grato a todos que, de alguma forma, contribuíram para este trabalho – com incentivo, revisões criteriosas ou simplesmente acreditando em meu potencial.

Por fim, o presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM — por meio do projeto POSGRAD.

A todos vocês, meu mais sincero obrigado!

"Os contos de fadas são mais do que verdadeiros: não porque nos dizem que os dragões existem, mas porque nos dizem que os dragões podem ser derrotados"

(C.K. Chesterton.)

Resumo

Com os avanços da inteligência artificial, especialmente em redes neurais profundas, diversas tarefas cotidianas vêm sendo executadas por esses modelos com precisão comparável ou superior à dos seres humanos. No entanto, essas redes enfrentam uma limitação crítica: a incapacidade de aprender continuamente sem degradar o desempenho em tarefas previamente aprendidas, fenômeno conhecido como esquecimento catastrófico, observado desde 1989. Diversas abordagens foram propostas para mitigar esse problema, classificadas em quatro grupos principais: i) rehearsal; ii) baseadas em distância; iii) sub-redes; e iv) redes dinâmicas. Dentre essas, destacam-se os métodos baseados em sub-redes e redes dinâmicas, sendo estas últimas particularmente promissoras por permitirem a expansão do modelo ao longo do tempo. Nesta tese, propomos o uso da ConvLoRA para aprendizado contínuo em redes convolucionais, inspirada nos adaptadores Low-Rank (LoRA) originalmente aplicados em modelos de linguagem natural.

A ConvLoRA introduz adaptadores treináveis específicos para cada nova tarefa em todas as camadas convolucionais, permitindo que o modelo aprenda incrementalmente sem sobrescrever os parâmetros anteriores e sem armazenar dados de tarefas passadas, respeitando, assim, restrições de privacidade. Quando combinada ao LoRA nas camadas totalmente conectadas, a abordagem resultou em um crescimento de apenas 2,2% no número total de parâmetros por tarefa. Em cenários que consideram apenas as camadas convolucionais, o uso exclusivo da ConvLoRA proporcionou uma redução de 85,2% no crescimento do modelo em relação a abordagens clássicas como o SeNA-CNN.

Os experimentos foram conduzidos em *benchmarks* consolidados de aprendizado contínuo para tarefas de classificação de imagens, incluindo os conjuntos CIFAR-100 e CUB-200, demonstrando que a combinação ConvLoRA + LoRA mantém ou se aproxima da acurácia média dos modelos de referência, com eficiência paramétrica significativamente superior. Esses resultados confirmam que a abordagem proposta é eficaz, escalável e adequada para aplicações práticas que exigem aprendizado contínuo sob restrições de memória e privacidade.

Abstract

With the advancement of artificial intelligence, particularly deep neural networks, several daily tasks are now being performed by these models with accuracy comparable to or even surpassing that of humans. However, such networks face a critical limitation: their inability to learn continuously without degrading performance on previously learned tasks—a phenomenon known as catastrophic forgetting, first identified in 1989. Several strategies have been proposed to mitigate this problem, generally classified into four groups: (i) rehearsal-based; (ii) distance-based; (iii) sub-network-based; and (iv) dynamic networks. Among these, sub-network and dynamic approaches stand out, with the latter showing greater promise due to their ability to expand the model over time. In this thesis, we propose the use of ConvLoRA for continuous learning in convolutional neural networks, inspired by Low-Rank Adapters (LoRA) originally applied in natural language processing models.

ConvLoRA introduces task-specific trainable adapters into every convolutional layer, enabling incremental learning without overwriting previous parameters and without storing data from past tasks, thereby adhering to data privacy constraints. When combined with LoRA in fully connected layers, the proposed approach results in only a 2.2% increase in total parameters per task. In scenarios focusing solely on convolutional layers, using ConvLoRA exclusively leads to an 85.2% reduction in model growth compared to traditional methods such as SeNA-CNN.

Experiments were conducted on established benchmark datasets for continual learning in image classification, including CIFAR-100 and CUB-200. Results demonstrate that the ConvLoRA + LoRA combination maintains or approaches the average accuracy of state-of-the-art methods while delivering significantly greater parameter efficiency. These findings confirm that the proposed method is effective, scalable, and suitable for real-world applications that require continuous learning under memory and privacy constraints.

Lista de Figuras

Lista de Tabelas

Acrónimos

CF Esquecimento catastrófico

AE *Autoencoder*

ANN Redes Neurais Artificiais (*Artificial Neural Networks - ANN*)

i.i.d. Independentes e Identicamente Distribuídas

IA Inteligência Artificial

MRE Minimização do Risco Empírico

RNC Redes Neurais Convolucionais (*Deep Convolutional Networks*)

RNP Redes Neurais Profundas - RNP (*Deep Neural Networks*)

SVM Máquinas de Vetores de Suporte (*Support Vector Machine*)

Nomenclatura

$\mathscr{L}$ Função de perda.

a Função de ativação.

D Conjunto de amostras.

l Camada de uma rede neural.

n_l Neuronio da camada l.

P Distribuição de probabilidades.

R Risco emprírico.

T Tarefa representada por um conjunto de classes.

t_n N-ésima tarefa

W Pesos ou parâmetros de um modelo.

Y Classe verdadeira das amostras..

F Diagonal da matriz de Fisher.

M Modelo de rede neural.

Sumário

1

Introdução

A criação de agentes inteligentes capazes de aprender e tomar decisões de forma similar aos seres humanos é um dos principais objetivos da pesquisa em Inteligência Artificial [Turing, 2004]. Modelos de IA que se assemelham ao funcionamento do cérebro humano são as Redes Neurais Artificiais (*Artificial Neural Networks - ANN*), compostas por neurônios artificiais interconectados que processam estímulos através de funções matemáticas, gerando respostas ou ações [Yegnanarayana, 2009]. Estes modelos são amplamente utilizados em problemas de classificação, onde cada estímulo deve ser categorizado em uma classe conhecida.

Contudo, a classificação de dados não estruturados, como imagens, áudios e textos, apresenta desafios significativos. Uma técnica para simplificar o aprendizado destes modelos é a utilização de características extraídas dos dados, processo conhecido como *feature engineering* [Turner et al., 1999]. A extração de características mais discriminativas facilita o aprendizado dos modelos, mas geralmente requer conhecimento especializado, o que pode limitar sua aplicabilidade a uma ampla gama de problemas e instituições com recursos limitados.

Com os avanços tecnológicos, tanto em poder computacional quanto na aquisição de grandes bases de dados, a extração de características pode ser incorporada diretamente nas ANN [Krizhevsky et al., 2012]. Esta evolução levou ao desenvolvimento das Redes Neurais Profundas - RNP (*Deep Neural Networks*), que possuem múltiplas camadas e são capazes de inferir automaticamente as características mais relevantes para a discriminação de classes [LeCun et al., 2015].

As redes neurais convolucionais (RNC) são um tipo especializado de redes neurais profundas (RNP), projetadas especificamente para processar dados estruturados espacialmente, como imagens. Diferentemente das redes neurais totalmente conectadas, as RNC utilizam camadas convolucionais para extrair automaticamente características hierárquicas

dos dados, capturando padrões locais e reduzindo a necessidade de engenharia manual de atributos. Essa propriedade torna as RNC altamente eficazes em tarefas como classificação, segmentação e detecção de objetos, sendo amplamente empregadas em aplicações de visão computacional.

Para que as RNP alcancem alta assertividade, são necessários alguns requisitos: i) um grande volume de amostras para cada classe; ii) a garantia de que as amostras sejam Independentes e Identicamente Distribuídas; iii) múltiplas iterações de treinamento; e iv) a correspondência entre as classes das amostras utilizadas no treinamento e aquelas a serem aprendidas na fase de inferência.

Entretanto, esses requisitos nem sempre são atendidos em problemas de classificação do dia a dia. A situação se torna ainda mais desafiadora quando o modelo precisa aprender novas classes ao longo do tempo, pois essas classes não estavam presentes no conjunto de treinamento original. Diferentemente de uma simples mudança na distribuição dos dados, a introdução de novas classes exige que o modelo expanda seu conhecimento sem esquecer o que já foi aprendido, garantindo que tanto as classes antigas quanto as recém-adicionadas sejam corretamente reconhecidas. Caso essa adaptação não seja realizada de forma adequada, o desempenho do modelo pode ser significativamente comprometido.

Portanto, para que uma RNP seja efetivamente utilizada em produção, é essencial que ela seja capaz de aprender novos conceitos ao longo do tempo, sem perder o conhecimento previamente adquirido. Esse processo é conhecido como aprendizado incremental, que consiste na expansão contínua do conhecimento do modelo para incorporar novas classes sem comprometer a capacidade de reconhecer as classes já aprendidas. Diferente de um simples ajuste nos pesos da rede para adaptar-se a variações nos dados, o aprendizado incremental exige que o modelo consiga armazenar e integrar novas categorias, garantindo que a base de conhecimento se amplie progressivamente [Diethe et al., 2019]. Apesar dos avanços recentes, ainda enfrenta desafios significativos, como o esquecimento catastrófico [McCloskey & Cohen, 1989], fenômeno no qual a incorporação de novos conhecimentos pode levar à perda de informações previamente adquiridas.

O esquecimento catastrófico é um fenômeno que ocorre principalmente em modelos parametrizados, onde ao alterar os seus parâmetros para aprender uma nova classe, perde significativamente sua capacidade de resolver as tarefas que previamente o modelo conseguia. Esse fenômeno pode ser abordado por meio de várias configurações: i) *Domain Incremental Learning*; ii) *Task Incremental Learning*; iii) *Class Incremental Learning*; iv) *Online Incremental Learning*; v) *Unbounded Task Incremental Learning*; e, vi) *Data-free Incremental Learning*. Cada configuração possui suas características e é relevante para diferentes contextos de aprendizagem contínua, detalhados no Seção 3.2.

O conceito de *Task Incremental Learning* é particularmente crucial para modelos que

precisam aprender novas tarefas, como é o caso de esteiras de produção em fábricas. Cada tarefa é um conjunto de novas classes que o modelo precisa aprender a classificar. Nessas situações, novos produtos (classes) são frequentemente introduzidos ao longo da vida útil do modelo, exigindo atualizações regulares sem comprometer o desempenho nas tarefas anteriores. Neste exemplo, cada tarefa é considerada um conjunto de classes de produto que o modelo deve ter a capacidade de discriminar. Um outro exemplo relevante é o de sistemas de reconhecimento facial em aeroportos, onde novos dados de indivíduos ou alterações nas características faciais devem ser constantemente incorporados para manter a segurança e eficiência do sistema.

A aprendizagem contínua abrange diversos contextos, entre eles a classificação de imagens [Singh et al., 2020], a detecção de objetos [Shmelkov et al., 2017], a segmentação de imagens [Cermelli et al., 2020], a geração de imagens [Zhai et al., 2021], a classificação de textos [Huang et al., 2021], o reconhecimento de entidades nomeadas [Monaikul et al., 2021], o aprendizado por reforço [Kaplanis et al., 2019] e os sistemas de recomendação [Mi et al., 2020b, Xu et al., 2020].

O contexto de classificação de imagens serve como base para muitos outros domínios visuais. Neste trabalho, focamos no problema de classificação de imagens na configuração *Task Incremental Learning*, fundamental para diversas aplicações práticas. Investigamos como uma RNP pode aprender novos conjuntos de classes (ou tarefas) visuais sem comprometer a qualidade das classes já conhecidas. Assim, as seguintes questões orientam esta pesquisa:

1. É possível que um modelo aprenda novas tarefas sequencialmente sem perder qualidade nas tarefas já aprendidas?
2. Existe um limite para o número de tarefas que podem ser aprendidas?
3. A ordem em que as tarefas são aprendidas interfere na retenção do conhecimento?
4. O modelo precisa ser expandido para poder aprender novas tarefas?

1.1 Motivação

Modelos de RNP têm demonstrado resultados equivalentes ou superiores aos humanos em diversas tarefas, como identificação de objetos em imagens [LeCun et al., 2015] e geração de áudio [Oord et al., 2016]. Contudo, a busca por maior acurácia e eficiência continua a impulsionar o desenvolvimento de novos modelos [Devlin et al., 2019, Tian et al., 2019]. Esses modelos são amplamente utilizados em nosso cotidiano, integrados a sistemas como reconhecimento facial para desbloqueio de celulares, recomendações em plataformas de varejo

e streaming, assistentes conversacionais inteligentes e personagens não jogáveis em jogos eletrônicos[1].

Entretanto, a aplicação desses modelos em cenários com incremento dinâmico de tarefas é limitada, pois eles não estão aptos a aprender continuamente. Na maioria das vezes, a atualização de um modelo exige a adição de novas amostras à base de dados, o retrainamento completo e a substituição da versão antiga, o que gera custos computacionais elevados e dificuldades de manutenção, especialmente com o aumento do número de tarefas e amostras armazenadas [Robins, 1993, 1995]. Além disso, esse processo de aprendizado incremental é vulnerável ao esquecimento catastrófico, onde a adaptação aos novos dados resulta na perda de desempenho em tarefas previamente aprendidas.

O desenvolvimento de modelos capazes de se adaptar a novas tarefas com menor custo computacional é, portanto, crucial. Modelos que podem se adaptar continuamente sem a necessidade de armazenar grandes quantidades de dados antigos ou de reprocessar todos os dados previamente vistos são altamente desejáveis. Esse avanço pode prolongar a vida útil dos sistemas e permitir personalizações, especialmente em dispositivos de borda.

Dessa forma, explorar e implementar técnicas que mitiguem o esquecimento catastrófico é essencial para desenvolver sistemas de aprendizado contínuo mais robustos e eficientes. Essas técnicas não apenas preservam a acurácia em tarefas antigas, mas também podem minimizar o crescimento do modelo e a demanda por recursos no aprendizado contínuo, facilitando a manutenção e a escalabilidade dos sistemas de inteligência artificial.

1.2 Problema do Esquecimento Catastrófico

As RNPs com a capacidade de aprendizado contínuo podem aprender novas tarefas de três formas: i) sequencial; ii) cumulativo; e, iii) por *stream*. No aprendizado sequencial, em cada sessão de treinamento uma nova tarefa é apresentada ao modelo, sem que o mesmo tenha acesso a exemplos de tarefas anteriores. No aprendizado cumulativo, cada sessão pode conter amostras de uma nova tarefa ou não. Por último, o aprendizado por *stream* é semelhante ao cumulativo, mas cada nova amostra é considerada uma tarefa. Diferentemente dos dois primeiros métodos, o aprendizado por *stream* exige ajuste *online*, ou seja, não há tempo extra para receber uma nova tarefa e executar várias iterações de treinamento.

A Figura 1.1 ilustra esse fluxo de aprendizado contínuo para sistemas de reconhecimento de imagens. Inicialmente, o modelo sabe reconhecer imagens de gato, pássaro e cachorro. Em seguida, novos objetos, como avião e barco, são apresentados ao modelo, que passa a reconhecer os cinco tipos de imagens. À medida que novos objetos são aprendidos,

[1] Um personagem não jogável (em inglês: *non-player character* ou NPC) é um personagem de jogo eletrônico que não pode ser controlado por um jogador.

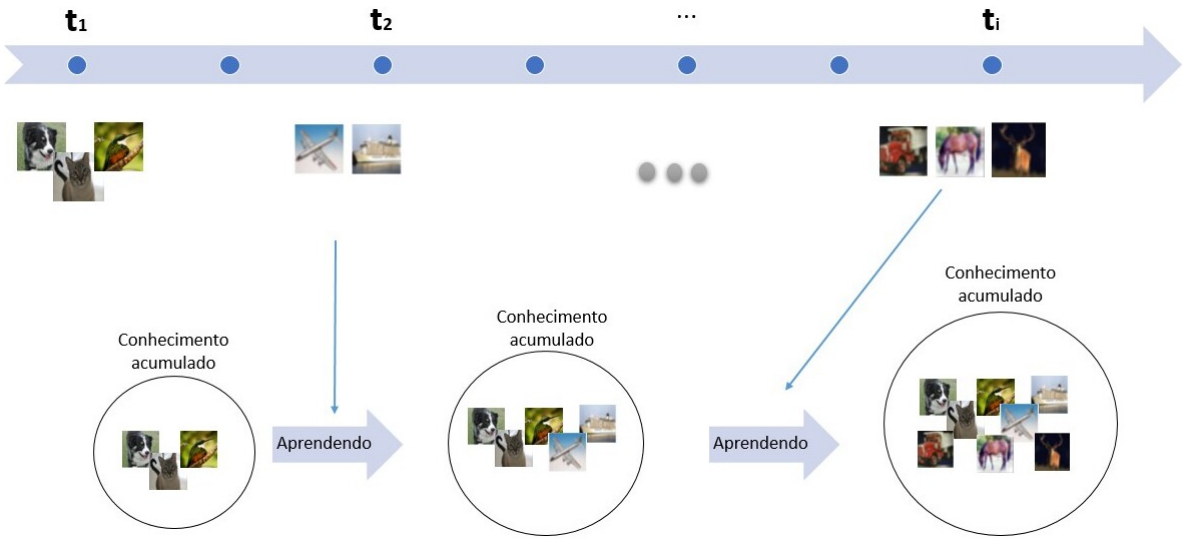


Figura 1.1. Fluxo geral do aprendizado contínuo. A cada intervalo, o modelo aprende a identificar novas imagens.

o conhecimento do modelo se expande. Embora este exemplo trate do reconhecimento de imagens, o fluxo pode ser estendido para outros tipos de tarefas.

Assim, podemos definir o problema da seguinte forma: dado um modelo $\mathbb{M}$ e um conjunto de tarefas $T = \{t_1, t_2, t_3, \ldots, t_n\}$, $\mathbb{M}$ deve aprender sequencialmente todas as tarefas contidas em T, sujeito às seguintes condições:

1. Após o modelo $\mathbb{M}$ aprender a tarefa t_p com $p \geq 2$, o desempenho em todas as tarefas anteriores t_j, para $1 \leq j < p$, não deve diminuir significativamente;

2. O modelo não pode armazenar todos os dados vistos em tarefas previamente aprendidas;

3. O aumento da quantidade de parâmetros deve ser, no máximo, proporcional ao crescimento linear do modelo.

Além das características comumente abordadas em trabalhos que visam mitigar o esquecimento catastrófico, há outras que variam conforme a configuração adotada. Nesta pesquisa, concentramos nossos esforços no *Task Incremental Learning*, no qual se pressupõe que, durante a avaliação, o modelo tem conhecimento prévio da tarefa à qual cada amostra pertence. Diferentemente do *zero-shot learning*, onde o modelo precisa inferir classes nunca vistas durante o treinamento, no aprendizado incremental cada nova classe adicionada ao modelo é previamente aprendida em uma tarefa (conjunto de classes) alguma etapa do processo de treinamento. Assim, o desafio central do *Task Incremental Learning* não é a generalização para classes desconhecidas, mas sim a retenção do conhecimento adquirido, evitando o esquecimento catastrófico ao longo da incorporação de novas tarefas.

Por exemplo, suponha que o modelo seja inicialmente treinado para classificar imagens entre gato e cachorro e, em seguida, aprenda uma nova tarefa para classificar imagens entre pássaros e zebras. Durante a inferência de uma amostra qualquer, o modelo sabe a qual tarefa ela pertence, ou seja, se faz parte da primeira ou da segunda tarefa.

Nesta configuração, cada tarefa é definida por um conjunto de classes a ser aprendido pelo modelo. Assim, definimos $T = \{t_i \mid i = 1, \ldots, |T|\}$ como um conjunto de tarefas a ser aprendido de forma incremental. Ao aprender t_i, o modelo tem acesso apenas aos dados de treinamento de t_i, mas, durante a sua avaliação, ele deve ser testado com todas as tarefas já conhecidas, ou seja, t_j para todo $j \leq i$.

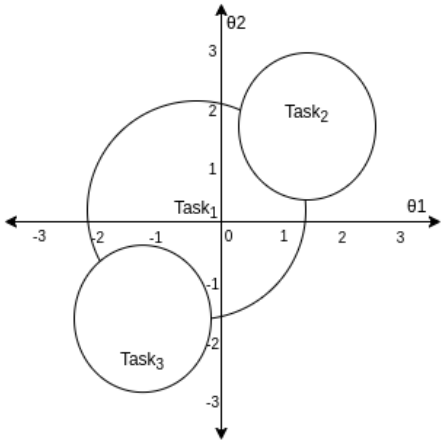
1.2.1 As limitações das RNPs

Apesar das RNPs serem modelos promissores para alcançar uma inteligência artificial similar à dos seres humanos, elas sofrem com o esquecimento catastrófico quando submetidas ao aprendizado contínuo sem mecanismos de retenção de conhecimento [Goodfellow et al., 2014a]. Esse desafio foi evidenciado por Knoblauch et al. [2020], que demonstrou que o problema de determinar os parâmetros ótimos para evitar o esquecimento catastrófico (CF) em um modelo de tamanho fixo pode ser reduzido ao conhecido problema da Satisfatibilidade (SAT) [Schaefer, 1978]. Como consequência, essa prova classifica o problema dentro da classe de problemas *NP-HARD*.

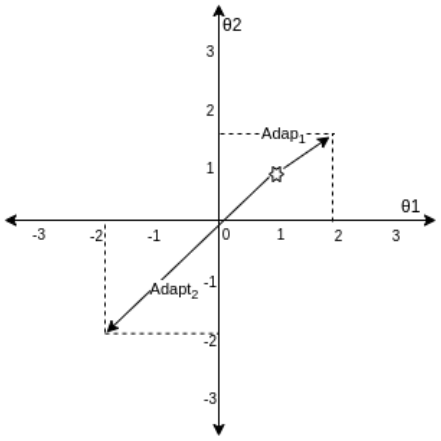
Para ilustrar as limitações de modelos não dinâmicos que compartilham os mesmos parâmetros em diferentes tarefas, considere um modelo simples M, cujo conjunto de parâmetros é representado por $\mathbf{W} \in \mathbb{R}^2$, de tamanho $|\mathbf{W}| = 2$, onde cada parâmetro w_i varia no intervalo $[-3, 3]$. Isso resulta em $49 = 7^2$ configurações distintas possíveis para o modelo. Além disso, suponha a existência de três tarefas, t_1, t_2 e t_3, cada uma associada a uma região específica dentro do espaço de parâmetros.

A tarefa t_1 pode ser resolvida pelo modelo M se a configuração dos parâmetros estiver dentro de uma região circular centrada em $(0,0)$ com raio igual a 2. Já a tarefa t_2 é solucionável quando a configuração dos parâmetros pertence a uma região circular menor, centrada em $(2,2)$ com raio igual a 1. Por fim, a tarefa t_3 pode ser resolvida quando os parâmetros estão dentro de outra região circular centrada em $(-2,-2)$, também com raio igual a 1.

A Figura 1.2(a) ilustra essas regiões de solução no espaço de parâmetros, onde cada círculo representa a área onde determinada tarefa pode ser resolvida. Como não há interseção entre essas regiões, nenhuma configuração única dos parâmetros pode satisfazer todas as tarefas simultaneamente. Assim, o modelo M não consegue encontrar um único conjunto de parâmetros W que resolva todas as tarefas ao mesmo tempo. Mesmo que o modelo tenha acesso aos dados de treinamento de todas as tarefas e passe por um treinamento extensivo,



((a)) Regiões de solução das tarefas no espaço de parâmetros. Cada círculo representa a região onde um conjunto de parâmetros resolve uma tarefa específica (t_1, t_2 ou t_3). Como não há interseção entre essas regiões, nenhuma configuração única pode resolver todas as tarefas simultaneamente.



((b)) Conceito de adaptadores dinâmicos. O ponto estrela representa o modelo resolvendo t_1. Adaptadores nos pontos $(1,1)$ e $(-3,-3)$ são aplicados para permitir a resolução de t_2 e t_3, respectivamente.

Figura 1.2. Configurações de parâmetros para resolver as tarefas e o conceito de adaptadores dinâmicos.

ele ainda enfrentará limitações de capacidade para lidar com os três desafios propostos.

A Figura 1.2(b) apresenta uma possível solução para esse problema: o conceito de adaptadores dinâmicos. Nesse caso, um ponto estrela representa a solução do modelo para a tarefa t_1, enquanto os adaptadores nos pontos $(1,1)$ e $(-3,-3)$ são aplicados para permitir que o modelo resolva t_2 e t_3, respectivamente. Essa abordagem ilustra como a utilização de componentes ajustáveis pode permitir que um mesmo modelo lide com múltiplas tarefas sem comprometer o desempenho global.

1.3 Abordagem Proposta

Diversas abordagens têm sido propostas na literatura para mitigar o problema do esquecimento catastrófico, geralmente com custos computacionais significativos, seja em termos de aumento do tamanho do modelo ou do tempo de treinamento. Para enfrentar esses desafios, propomos uma solução baseada na técnica de adaptadores *Low-Rank* (LoRA) e sua variante para camadas convolucionais, chamada ConvLoRA [Aleem et al., 2024].

A técnica de adaptadores busca mitigar o esquecimento catastrófico com um aumento mínimo no número de parâmetros e sem armazenar amostras de tarefas já aprendidas. Diferente das abordagens tradicionais que aumentam significativamente o número de parâmetros

ou armazenam grandes quantidades de dados antigos, nossa abordagem adiciona uma pequena quantidade de parâmetros específicos para cada nova tarefa, mantendo a precisão nos dados já aprendidos. Os adaptadores consistem em dois componentes principais: tensor **A**; e, **B**.

Esses tensores quando multiplicadas geram um novo tensor intermediária de mesma cardinalidade que **W** (tensor de parâmetros original da camada do modelo). Assim, podemos somar os pesos de **W** com o tensor intermediário elemento-a-elemento, gerando uma adaptação ao modelo sem necessariamente afetar os parametros originais do mesmo, conforme equação abaixo:

$$\mathbf{W}_{novo} = \mathbf{W} + \mathbf{A} \cdot \mathbf{B},$$

sendo $\mathbf{W}_{\text{novo}} \in \mathbb{R}^{k \times k}$ os parâmetros da camada adaptados para a nova tarefa, $\mathbf{W} \in \mathbb{R}^{k \times k}$ os parâmetros ajustados para a tarefa inicial, e $\mathbf{A} \in \mathbb{R}^{k \times d}$ e $\mathbf{B} \in \mathbb{R}^{d \times k}$ as componentes do adaptador, onde d é a dimensão do adaptador. Dessa forma, a quantidade de novos parâmetros é significativamente menor quando comparada ao modelo original completo.

As camadas de normalização do modelo (*Batch Normalization*) são duplicadas a cada nova tarefa. Isso garante que as estatísticas das novas tarefas não interfiram nas tarefas previamente aprendidas, conforme descrito em Pham et al. [2022]. O aumento no número de parâmetros causado pela duplicação dessas camadas pode ser considerado irrelevante quando comparado ao total de parâmetros do modelo.

Embora nossa proposta resulte em um aumento no tamanho do modelo, categorizando-se assim no grupo de redes dinâmicas (como será discutido no Capítulo 3), entendemos que essa abordagem é essencial para permitir que um modelo aprenda tarefas conflitantes em termos de solução.

Nossa metodologia é avaliada em conjuntos de dados padrão para aprendizado contínuo, demonstrando que os adaptadores podem mitigar o esquecimento catastrófico com um aumento de pouco mais de 2% no número de parâmetros para cada nova tarefa. Além disso, os adaptadores preservam a precisão em tarefas antigas e eliminam a necessidade de armazenar dados anteriores, conforme evidenciado pelos resultados experimentais.

Os principais benefícios dos adaptadores podem ser resumidos nos seguintes pontos:

- **Redução no aumento do tamanho do modelo**: o uso de adaptadores permite uma redução significativa no crescimento do tamanho do modelo, adicionando aproximadamente 2% de parâmetros treináveis ao modelo VGG19 original;

- **Eliminação da necessidade de armazenamento de dados antigos**: nossa abordagem remove a necessidade de armazenar dados anteriores, um requisito comum em muitas

técnicas existentes para aprendizado contínuo, aumentando assim a escalabilidade e a conformidade com restrições de privacidade de dados;

- **Preservação da acurácia**: a modularidade do LoRA e do ConvLoRA garante que a acurácia em tarefas antigas seja preservada, com nossa abordagem superando empiricamente os métodos de referência em termos de desempenho em cenários de aprendizado contínuo.

Em suma, os adaptadores oferecem uma solução eficiente e escalável para o problema do esquecimento catastrófico em redes neurais convolucionais, com potencial para aplicação em diversos cenários de aprendizado contínuo.

1.4 Objetivos

O objetivo principal de nosso trabalho é combater ou reduzir os efeitos do esquecimento catastrófico no aprendizado incremental de tarefas no contexto de classificação de imagens usando redes neurais convolucionais (RNC), por meio de adaptadores *Low-Rank* (*Low-Rank Adapters* e *Convolutional-Low-Rank Adapters*).

Para atingir este objetivo, os seguintes objetivos específicos devem ser alcançados:

- Avaliar empiricamente a capacidade de uma RNC pré-treinada em lidar com tarefas incrementais sem técnicas de mitigação;
- Desenvolver e implementar adaptadores *Low-Rank* e *Convolutional-Low-Rank* integrados às camadas de uma RNP;
- Medir o impacto dos adaptadores em termos de acurácia e crescimento do modelo em *benchmarks* de classificação incremental;

Ao atingir esses objetivos, esperamos contribuir significativamente para o avanço do estado da arte no aprendizado contínuo, oferecendo uma solução prática e eficiente para o problema do esquecimento catastrófico em redes neurais convolucionais durante o aprendizado de novas tarefas.

1.5 Considerações Finais do Capítulo

Neste capítulo, apresentamos uma introdução à evolução do campo de aprendizado de máquinas com o advento das RNP. Embora esse campo tenha avançado rapidamente, o aprendizado contínuo permanece um desafio significativo para a aplicação desses modelos em

ambientes dinâmicos. Discutimos o problema do esquecimento catastrófico, um dos principais obstáculos ao aprendizado contínuo desses modelos. Classificamos esse aprendizado em seis configurações distintas, cada uma direcionada a diferentes contextos de aplicação. Dentro desse cenário, levantamos questões cruciais que orientam esta pesquisa. Por fim, propomos uma solução específica para mitigar o problema do esquecimento catastrófico.

As principais contribuições esperadas desta tese incluem:

- Definição de uma taxonomia para as abordagens existentes;
- Desenvolvimento de uma nova abordagem para o aprendizado contínuo no contexto de classificação de imagens na configuração de aprendizado incremental, capaz de superar os problemas identificados.

A estrutura deste trabalho está organizada da seguinte forma: no Capítulo 2, apresentamos os fundamentos e conceitos teóricos relacionados ao esquecimento catastrófico em RNC. No Capítulo 3, realizamos uma análise detalhada do problema e propomos uma taxonomia para classificar os estudos existentes na área. No Capítulo 4, introduzimos nossa abordagem para mitigar o esquecimento catastrófico em modelos de RNC. Os resultados experimentais são discutidos no Capítulo 5, avaliando o desempenho da proposta em diferentes cenários. Por fim, no Capítulo 6, apresentamos as conclusões e discutimos possíveis direções para trabalhos futuros.

2

Fundamentos Teóricos

Neste capítulo apresentamos os conceitos necessários para compreender os modelos de RNP e como estes são afetados pelo esquecimento catastrófico. Esses conceitos são fundamentais para o entendimento do Capítulo 4, onde apresentamos a proposta desta tese.

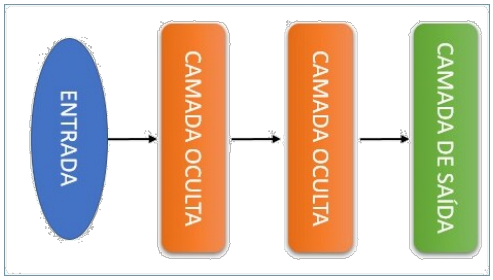
2.1 Redes Neurais

Redes neurais artificiais são compostas por camadas de um ou mais neurônio [Zurada, 1992]. Elas possuem uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída [Norvig & Russell, 2004]. Os dados de um problema é inserido à camada de entrada. Os neurônios de uma camada l_i processam o estimulo[1] e aplica uma função de ativação para produzir uma resposta. Essa resposta é usada para estimular a camada l_{i+1}, até o estimulo chegar a camada de saída. A Figura 2.1(a) apresenta uma arquitetura básica das redes neurais. Existem varias outras arquiteturas[2] que extrapolam o exemplo da figura, entretanto, fogem do escopo desta tese.

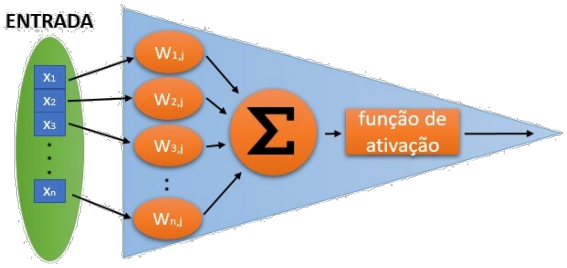
Cada neurônio n_l da camada l é responsável por aprender parte de um padrão e a saída do neurônio é composta por uma soma ponderada de suas entradas sujeito a uma função de ativação no final (o que torna o modelo não linear). Os pesos, também chamados de parâmetros, de cada neurônio é o que define o padrão aprendido pelo mesmo. Tomando como exemplo o neurônio da Figura 2.1(b), se o peso W_{2j} for próximo de zero (0) significa que a entrada x_2 é pouco relevante para o padrão, pois independente do valor de x_2 sua contribuição para o somatório após a multiplicação por W_{2j} será insignificante.

[1] Os dados de um problema ou o produto gerado por um neurônio é conhecido como estímulo.

[2] `https://www.asimovinstitute.org/neural-network-zoo/`



((a)) Arquitetura básica de redes neurais artificiais.



((b)) Neurônio de uma rede neural artificial.

Figura 2.1. Ilustração de uma rede neural profunda e um neurônio artificial.

Uma característica inerente desses modelos é a sua sensibilidade aos pesos dos neurônios. Uma alteração nesses pesos pode fazer um modelo identificar outros padrões. Isso pode fazer com que um modelo perca sua precisão em uma tarefa. Apesar desta sensibilidade, a alteração nos pesos ocorre sempre que o modelo processa uma amostra no modo de treinamento. Este ajuste nos pesos ocorre para que o modelo fique mais assertivo para aquela amostra.

O objetivo do treinamento de um modelo é encontrar um conjunto de pesos que permita classificar corretamente todas as amostras. No entanto, como nem todas as possíveis amostras podem ser conhecidas previamente, a distribuição real dos dados, representada por $P(D_k,k)$, onde d são as amostras e k suas respectivas classes, não é acessível a priori. Para lidar com essa limitação, adota-se uma aproximação chamada risco empírico [Vapnik, 1991], que deve ser minimizado para maximizar a capacidade de generalização do modelo (MRE), conforme a Equação 2.1.

Seja M o modelo em aprendizado, R o risco empírico, $\mathscr{L}$ a função de perda, W os pesos de M, D o conjunto de amostras disponíveis para treinamento e $Y[\cdot]$ o mapeamento de cada amostra para sua respectiva classe. Assim, temos:

$$\arg\min_{W} R(M) = \frac{1}{|D|}\sum_{i=1}^{|D|}\mathscr{L}(M(d_i),Y[d_i]), \text{ sujeito a } \begin{cases} 1 & \text{se } (M(d_i)=Y[d_i]) \\ 0 & \text{caso contrário} \end{cases} \tag{2.1}$$

Um neurônio, denotado por n_l, é matematicamente representado como:

$$n_l(in) = a\left(\sum_{n=1}^{|W|} w_n \cdot in_n\right), \tag{2.2}$$

onde in representa os estímulos de entrada do neurônio, W o conjunto de pesos associados e a a função de ativação. Há diversas funções possíveis para a, como a *sigmoid*, que res-

tringe a saída ao intervalo $[0,1]$; a tangente hiperbólica, que força os valores a ficarem entre $[-1,1]$; e a *Rectified Linear Unit* (ReLU), que zera os valores negativos. A ReLU, em particular, possibilitou grandes avanços no treinamento de modelos baseados em redes neurais profundas, pois atenuou os efeitos do desaparecimento do gradiente [Hochreiter et al., 2001], permitindo o desenvolvimento de arquiteturas mais complexas e eficientes.

2.1.1 Redes Neurais Profundas

Quando os modelos RNP apresentam muitas camadas interligadas são chamados de redes neurais profundas [Krizhevsky et al., 2012] e conseguem aprender composições de padrões complexas que fazem o modelo superar um ser humano em tarefas específicas [He et al., 2015]. Diversas áreas já utilizam esses modelos profundos para resolver problemas específicos, como na área da medicina [Huynh et al., 2016, Lu et al., 2017, Pham et al., 2016] e jogos [Clark & Storkey, 2015, Van Hasselt et al., 2016].

As primeiras camadas de uma rede tende a aprender as características simples encontradas no espaço de entrada. Por exemplo, em problemas de visão computacional, onde a entrada são imagens, essas características simples podem ser arestas, cores, contrastes, entre outros. As últimas camadas são responsáveis por juntar as características simples, formando representações mais complexas, como um nariz, uma boca ou um olho [Zacarias & Alexandre, 2018b].

No contexto de visão computacional, essas camadas são chamadas de convolucionais [Krizhevsky et al., 2012]. O objetivo dessas camadas é aprender os filtros convolutivos. Essas camadas são essenciais para trabalhar com problemas que o estímulo de entrada apresenta uma grande dimensão, pois reduz a quantidade de parâmetros do modelo. As representações geradas por esses filtros são chamadas de mapas de *features*. No final de uma rede convolucional, é necessário converter os mapas de *features* em um vetor unidimensional para poder gerar uma classificação, formando camadas completamente conectadas.

Nessas camadas completamente conectadas, as primeiras buscam combinar o estímulo de entrada para projeta-las em um espaço $\mathbb{R}^n$ de n dimensões de tal forma que as representações sejam mais discrimináveis. Dessa forma, a cada camada, espera-se que seja mais fácil fazer a classificação desses estímulos. Essas representações são chamadas de vetor de características, ou simplesmente *embeddings*.

É comum os pesquisadores dividirem uma RNP em duas partes: i) um extrator de *embeddings* $\mathscr{E}$; e, ii) um classificador $\mathscr{Q}$. $\mathscr{E}$ é formado pelas camadas iniciais da RNP e o $\mathscr{Q}$ pelas camadas restantes. O ponto de separação depende da complexidade do problema, o objetivo é que $\mathscr{E}$ possa projetar as amostras do problema no espaço de *embedding* de tal forma que as classes sejam facilmente discrimináveis. Algumas vezes, é adotado um outro

classificador (diferente das classes restantes da RNP) como $\mathscr{2}$, por exemplo um SVM.

A Figura 2.2, do lado esquerdo, mostra que um problema $f(x) = x$, onde x é o estimulo de uma amostra (um atributo dos círculos e quadrados) não é linearmente discriminável, isso é, não existe uma linha reta que possa separar os dois grupos. No entanto, quando projetado em um espaço $\mathbb{R}^2$ com uma projeção $p = (y, y^2)$, sendo $y = f(x)$, fica trivial fazer essa discriminação, como mostra no lado direito da figura.

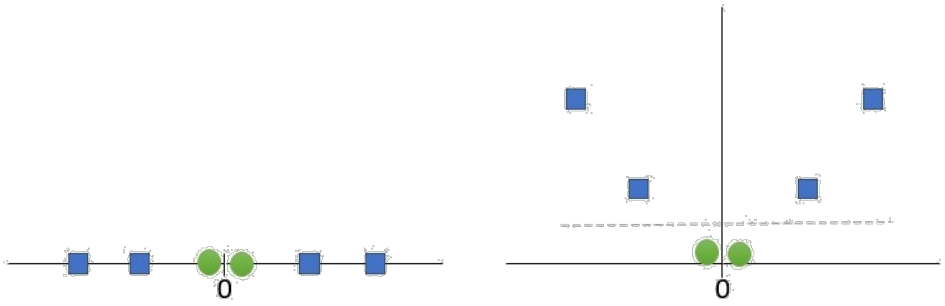


Figura 2.2. Função de projeção em espaço de representação. Do lado esquerdo as amostras (quadrados e círculos) são apresentadas de acordo com seu atributo em $\mathbb{R}^1$, onde não são linearmente separáveis. Do lado direito, as amostras são projetadas no espaço $\mathbb{R}^2$ de tal forma que é possível discrimina-las por uma reta (em cinza na imagem).

As RNP podem ser usadas também como modelos generativos, como autocodificadores (AE) [Ronneberger et al., 2015] e redes generativas adversarias (GAN) [Goodfellow et al., 2014b]. Esses modelos tem a capacidade de, partindo de uma representação no espaço $\mathbb{R}^n$ gerar uma amostra de um determinado problema. Por exemplo, partindo de um vetor de característica em um espaço de representações, assumindo valores aleatórios de uma distribuição normal, é possível gerar uma imagem complexa de um rosto humano ou uma paisagem [Aggarwal et al., 2021].

Independente da arquitetura, quando uma RNP aprende as amostras de forma sequencial (por classe), existe uma grande chance dele perder acurácia para as amostras das primeiras classes e ganhar para as últimas. Esse fenômeno é chamado de esquecimento catastrófico e é discutido na próxima seção.

2.1.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (RNC) foram desenvolvidas para lidar com dados estruturados em grade, como imagens, onde a informação local é fundamental. Diferente das redes neurais totalmente conectadas, que tratam cada entrada de forma independente, as RNC exploram a estrutura espacial dos dados através de operações matemáticas que capturam padrões locais. Essa abordagem é inspirada na organização do córtex visual de mamíferos,

onde diferentes neurônios respondem a estímulos específicos dentro de uma região do campo visual [Aloysius & Geetha, 2017].

O princípio fundamental das RNC está na operação de convolução. Formalmente, considerando uma entrada $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, onde H, W e C representam a altura, largura e número de canais da imagem, respectivamente, a convolução com um filtro $\mathbf{K} \in \mathbb{R}^{k \times k \times C}$ é definida como:

$$\mathbf{Y}_{i,j} = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \sum_{c=0}^{C-1} \mathbf{X}_{i+m,j+n,c} \cdot \mathbf{K}_{m,n,c}, \tag{2.3}$$

onde $\mathbf{Y} \in \mathbb{R}^{(H-k+1) \times (W-k+1)}$ representa o mapa de características gerado. A operação de convolução pode ser entendida como a aplicação de um conjunto de filtros, ou *kernels*, que extraem padrões locais da imagem. Inicialmente, esses padrões podem ser simples, como bordas e texturas, mas ao longo das camadas esses filtros aprendem representações mais complexas, como partes de objetos e até objetos completos [Zeiler & Fergus, 2014].

Para evitar a redução excessiva da dimensionalidade e preservar informações da borda, é comum adicionar preenchimento (*padding*), de modo que a saída tenha o mesmo tamanho que a entrada. Além disso, a convolução pode ser aplicada com um passo (*stride*) maior que 1, o que reduz a resolução da saída e captura padrões em escalas diferentes.

Após as camadas convolucionais, é comum utilizar camadas de *pooling*, que reduzem a dimensionalidade espacial enquanto mantêm as informações mais relevantes. O *max pooling*, por exemplo, é definido como:

$$\mathbf{Y}_{i,j} = \max_{(m,n) \in \Omega} \mathbf{X}_{i+m,j+n}, \tag{2.4}$$

onde Ω representa a região local de tamanho $s \times s$ sobre a qual a operação é realizada. Esse processo reduz a quantidade de parâmetros da rede e aumenta a invariância a pequenas translações.

A arquitetura de uma RNC típica pode ser dividida em duas partes principais: um extrator de características e um classificador. O extrator de características consiste nas camadas convolucionais e de *pooling*, responsáveis por transformar a entrada em um espaço latente de menor dimensionalidade, porém, mais discriminativo. Já o classificador é formado por camadas totalmente conectadas que transformam esse espaço latente em probabilidades associadas às classes. Supondo que a saída final da rede seja um vetor $\mathbf{z} \in \mathbb{R}^N$, onde N é o número de classes, a predição da rede é dada pela função *softmax*:

$$p(y = k|\mathbf{z}) = \frac{\exp(z_k)}{\sum_{j=1}^{N} \exp(z_j)}, \tag{2.5}$$

onde $p(y=k|\mathbf{z})$ representa a probabilidade da entrada pertencer à classe k. O treinamento das RNC é realizado minimizando uma função de perda, geralmente a entropia cruzada, definida como:

$$\mathscr{L} = -\sum_{k=1}^{N} y_k \log p(y=k|\mathbf{z}), \tag{2.6}$$

onde y_k representa a variável indicadora da classe correta.

Redes convolucionais profundas, como AlexNet [Krizhevsky et al., 2012], VGGNet [Simonyan & Zisserman, 2015] e ResNet [He et al., 2016], mostraram um desempenho excepcional em tarefas de visão computacional, superando humanos em vários desafios [He et al., 2015]. No entanto, esses modelos sofrem com o fenômeno do esquecimento catastrófico quando submetidos a cenários de aprendizado contínuo [McCloskey & Cohen, 1989].

2.1.3 Arquitetura VGG

A arquitetura VGGNet [Simonyan & Zisserman, 2015] tornou-se um marco no aprendizado profundo devido à sua simplicidade e eficácia na classificação de imagens. Originalmente desenvolvida para a competição ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a VGGNet demonstrou desempenho superior ao de modelos anteriores, destacando-se pelo uso de múltiplas camadas convolucionais pequenas. Além da classificação de imagens, essa arquitetura também é aplicada em outras tarefas de visão computacional, como segmentação semântica e detecção de objetos. Entretanto, nesta seção, focaremos exclusivamente em seu uso para classificação de imagens.

A VGGNet segue um design padronizado composto por blocos convolucionais intercalados com camadas de *pooling*, seguidos por camadas totalmente conectadas responsáveis pela classificação. A principal inovação da VGGNet foi a adoção exclusiva de filtros 3×3 nas camadas convolucionais, contrastando com modelos anteriores que utilizavam filtros maiores. O uso de filtros menores empilhados permitiu uma extração de características mais eficiente, capturando padrões complexos na imagem ao longo das camadas mais profundas. Cada camada convolucional é seguida por uma ativação ReLU (*Rectified Linear Unit*), que introduz não-linearidade ao modelo e acelera a convergência do treinamento.

A arquitetura apresenta uma hierarquia de camadas bem definida. Primeiramente, a imagem de entrada passa por um conjunto de camadas convolucionais que extraem características de baixa e média complexidade. Em seguida, as camadas de *Max Pooling*, com janelas 2×2, reduzem progressivamente a resolução espacial da imagem, preservando as informações mais relevantes para a classificação. A parte final da rede é composta por camadas totalmente conectadas, onde as duas primeiras possuem 4096 neurônios cada, seguidas

de uma camada final que contém N neurônios, correspondentes ao número de classes do problema. A saída é processada por uma função *softmax*, convertendo os valores em probabilidades associadas a cada classe.

A Tabela 2.1 apresenta a estrutura das duas variantes mais conhecidas da VGGNet, a VGG-16 e a VGG-19, que diferem apenas no número de camadas convolucionais em seus blocos intermediários. Como pode ser observado, ambas as versões compartilham a mesma organização geral, com um aumento no número de convoluções na VGG-19.

Tabela 2.1. Estrutura das arquiteturas VGG-16 e VGG-19

Camada	VGG-16	VGG-19
Conv3-64	2x	2x
MaxPool	1x	1x
Conv3-128	2x	2x
MaxPool	1x	1x
Conv3-256	3x	4x
MaxPool	1x	1x
Conv3-512	3x	4x
MaxPool	1x	1x
Conv3-512	3x	4x
MaxPool	1x	1x
Fully Connected	3 camadas	3 camadas
Softmax	1x	1x

Na Figura 2.3, apresentamos uma representação esquemática da arquitetura VGG-16. Como ilustrado, a rede inicia com uma entrada de imagem que percorre sucessivos blocos convolucionais e *pooling*, culminando nas camadas totalmente conectadas responsáveis pela decisão final. Esse design estruturado permite que a VGGNet aprenda representações robustas e escaláveis para tarefas de classificação.

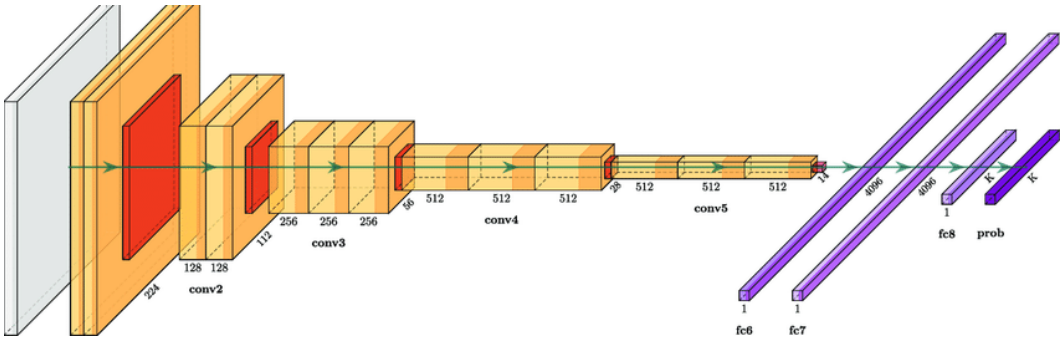


Figura 2.3. Representação da arquitetura VGG-16, destacando os blocos convolucionais, camadas de *pooling* e camadas totalmente conectadas. Figura de Blauch et al. [2020].

A popularidade da VGGNet na classificação de imagens deve-se à sua capacidade de generalização e ao seu design modular, que facilita a adaptação a diferentes bases de dados.

Apesar de seu sucesso, algumas limitações surgiram com o tempo, como o elevado número de parâmetros e o alto custo computacional. Ainda assim, a VGGNet continua sendo uma referência para o desenvolvimento de arquiteturas modernas de redes neurais convolucionais, sendo amplamente utilizada como base para outros modelos mais eficientes.

2.2 Aprendizado Contínuo

O aprendizado contínuo representa um paradigma emergente em *machine learning*, cujo objetivo é desenvolver sistemas capazes de aprender de forma incremental a partir de um fluxo contínuo de dados, sem a necessidade de re-treinamento completo a cada nova tarefa. Essa abordagem é especialmente relevante para redes neurais profundas, que historicamente sofrem com o problema do *esquecimento catastrófico* – a perda abrupta do conhecimento previamente adquirido quando expostos a novas classes [French, 1999]. Em ambientes dinâmicos, onde a novas tarefas (conjunto classes) podem surgir ao longo do tempo, os métodos tradicionais de treinamento, baseados em conjuntos de dados estáticos, revelam-se insuficientes para capturar as nuances e a evolução dos dados reais.

Formalmente, considere uma sequência de tarefas $\{T_1, T_2, \ldots, T_t\}$, em que cada tarefa T_t está associada a um conjunto de dados $\mathscr{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t}$. O objetivo é ajustar os parâmetros θ de um modelo de rede neural de forma que a função de perda acumulada $\min_\theta \sum_{t=1}^{n} \mathscr{L}_t(\theta)$, seja minimizada. Para cada tarefa, a função de perda é definida como:

$$\mathscr{L}_t(\theta) = \frac{1}{N_t} \sum_{i=1}^{N_t} \ell\big(f(x_i^t;\theta), y_i^t\big), \tag{2.7}$$

onde $f(x_i^t;\theta)$ representa a saída da rede para a entrada x_i^t e $\ell(\cdot,\cdot)$ é uma função de perda adequada (por exemplo, entropia cruzada ou erro quadrático) [Goodfellow et al., 2016]. A atualização dos parâmetros, utilizando métodos de descida de gradiente, é dada por:

$$\nabla_\theta \mathscr{L}_t(\theta) = \frac{1}{N_t} \sum_{i=1}^{N_t} \nabla_\theta \ell\big(f(x_i^t;\theta), y_i^t\big). \tag{2.8}$$

Entre as estratégias propostas para mitigar o esquecimento catastrófico, destaca-se a regularização dos parâmetros. Um dos métodos mais conhecidos é o *Elastic Weight Consolidation* (EWC), que introduz um termo de penalização na função de perda para restringir a alteração dos parâmetros essenciais às tarefas anteriores. A função de perda para uma nova

tarefa, considerando essa abordagem, é formulada como:

$$\mathscr{L}(\theta) = \mathscr{L}_{\text{novo}}(\theta) + \frac{\lambda}{2}\sum_i F_i\,(\theta_i - \theta_i^*)^2, \tag{2.9}$$

onde θ^* representa os parâmetros otimizados anteriormente, F_i é a diagonal da matriz de Fisher (que quantifica a importância dos parâmetros) e λ é um hiperparâmetro que controla o grau de restrição [Kirkpatrick et al., 2017]. Essa abordagem matemática evidencia que nem todos os parâmetros têm a mesma relevância para as tarefas já aprendidas, permitindo ao modelo preservar informações cruciais.

Outra estratégia relevante é o uso de técnicas de *replay*. Esse método consiste em armazenar e reintroduzir, durante o treinamento de novas tarefas, um subconjunto representativo dos dados já vistos. Ao combinar o conjunto de dados atual, $\mathscr{D}_{\text{novo}}$, com um subconjunto armazenado, $\mathscr{D}_{\text{replay}}$, a função de perda conjunta pode ser definida por:

$$\mathscr{L}(\theta) = \alpha\,\mathscr{L}\big(\theta; \mathscr{D}_{\text{novo}}\big) + (1-\alpha)\,\mathscr{L}\big(\theta; \mathscr{D}_{\text{replay}}\big), \tag{2.10}$$

sendo $\alpha \in [0,1]$ um parâmetro que equilibra o aprendizado entre novas informações e a retenção do conhecimento prévio [Rebuffi et al., 2017]. Essa técnica tem demonstrado eficácia tanto em estudos teóricos quanto empíricos, contribuindo para a manutenção do desempenho em tarefas previamente aprendidas.

Uma abordagem eficaz para lidar com o problema do esquecimento catastrófico no aprendizado contínuo consiste na expansão dinâmica do modelo. A ideia central é que, ao introduzir novos parâmetros ou módulos para acomodar tarefas futuras, evita-se que as atualizações necessárias para aprender novas informações interfiram diretamente nos parâmetros previamente otimizados para tarefas antigas.

Matematicamente, considere um modelo neural cuja função de predição pode ser escrita como $f(x;\theta) = f\big(x;\theta_{\text{base}}\big)$, onde θ_{base} representa o conjunto de parâmetros treinados até um certo ponto, responsável por modelar as tarefas anteriores. Ao enfrentar uma nova tarefa, a estratégia de expansão propõe a adição de um conjunto de novos parâmetros, θ_{novo}, de forma que o novo modelo seja dado por $f(x;\theta_{\text{base}},\theta_{\text{novo}}) = f_{\text{base}}(x;\theta_{\text{base}}) + \Delta f(x;\theta_{\text{novo}},\theta_{\text{base}})$, em que Δf é uma função adicional que depende dos novos parâmetros e, possivelmente, de θ_{base}. Em muitas arquiteturas dinâmicas, os parâmetros θ_{base} são mantidos fixos ou atualizados com regularização severa, enquanto θ_{novo} é livre para se ajustar à nova tarefa. Essa decomposição tem duas consequências importantes:

1. **Aumento da Capacidade de Representação:** A expansão adiciona novos graus de liberdade ao modelo. Do ponto de vista da teoria da aproximação, se o modelo original

com capacidade M aproxima funções dentro de uma certa classe, a expansão para uma capacidade $M + \Delta M$ aumenta o espaço hipótético, permitindo a aproximação de funções mais complexas ou que apresentam variações não capturadas anteriormente. Isso é compatível com o teorema da aproximação universal, que garante que redes neurais com uma quantidade suficiente de neurônios podem aproximar qualquer função contínua em um conjunto compacto.

2. **Desacoplamento dos Gradientes:** Durante o treinamento para uma nova tarefa, a atualização dos parâmetros é obtida via descida de gradiente. Se o modelo expandido é treinado com a seguinte função de perda:

$$\mathscr{L}(\theta_{\text{base}}, \theta_{\text{novo}}) = \mathscr{L}_{\text{antiga}}\big(\theta_{\text{base}}\big) + \mathscr{L}_{\text{nova}}\big(\theta_{\text{base}}, \theta_{\text{novo}}\big) + \lambda R(\theta_{\text{novo}}),$$

onde $\mathscr{L}_{\text{antiga}}$ é uma perda que assegura a preservação do desempenho nas tarefas anteriores (por exemplo, por meio de uma penalização ou de um replay) e $\lambda R(\theta_{\text{novo}})$ é um termo regularizador para os novos parâmetros, então a estratégia de congelar ou regularizar fortemente θ_{base} implica que os gradientes principais para a nova tarefa incidem sobre θ_{novo}. Assim, a atualização:

$$\theta_{\text{novo}} \leftarrow \theta_{\text{novo}} - \eta \, \nabla_{\theta_{\text{novo}}} \mathscr{L}(\theta_{\text{base}}, \theta_{\text{novo}})$$

é realizada sem afetar significativamente θ_{base}. Esse desacoplamento minimiza a interferência entre tarefas, permitindo que o conhecimento prévio seja preservado enquanto o modelo se adapta a novos dados.

Frameworks como os *Progressive Neural Networks* [Rusu et al., 2016] exemplificam essa abordagem, onde para cada nova tarefa é adicionado um "coluna"com novos parâmetros que se conectam lateralmente às colunas já existentes. De forma semelhante, o método das *Redes Dinâmicas Expandíveis* [Yoon et al., 2018] adapta a capacidade do modelo de forma incremental, adicionando neurônios ou camadas quando necessário, conforme critérios baseados na magnitude dos erros ou na ineficiência dos recursos atuais para modelar a nova tarefa.

2.3 Fatoração de Matrizes

A fatoração de matrizes é uma técnica essencial na álgebra linear e tem amplas aplicações em aprendizado de máquina, especialmente na redução de dimensionalidade, recomendação de sistemas e análise de dados estruturados.

Seja uma matriz $\mathbf{A} \in \mathbb{R}^{m \times n}$, a fatoração consiste em expressá-la como um produto de matrizes mais simples que preservam determinadas propriedades. Uma das decomposições mais conhecidas é a Decomposição em Valores Singulares (SVD), que expressa $\mathbf{A}$ como:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \tag{2.11}$$

onde $\mathbf{U} \in \mathbb{R}^{m \times m}$ é uma matriz ortogonal cujas colunas são os autovetores de $\mathbf{A}\mathbf{A}^T$; $\Sigma \in \mathbb{R}^{m \times n}$ é uma matriz diagonal com os valores singulares de $\mathbf{A}$; $\mathbf{V} \in \mathbb{R}^{n \times n}$ é uma matriz ortogonal cujas colunas são os autovetores de $\mathbf{A}^T\mathbf{A}$.

A decomposição SVD é amplamente utilizada para compressão de dados e Análise de Componentes Principais (PCA), permitindo a projeção de dados de alta dimensão em um espaço de menor dimensão, minimizando a perda de informação [Bishop, 2006].

Outra fatoração relevante é a Decomposição QR, que expressa $\mathbf{A}$ como o produto de uma matriz ortogonal $\mathbf{Q}$ e uma matriz triangular superior $\mathbf{R}$:

$$\mathbf{A} = \mathbf{Q}\mathbf{R} \tag{2.12}$$

onde $\mathbf{Q} \in \mathbb{R}^{m \times m}$ satisfaz $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$; $\mathbf{R} \in \mathbb{R}^{m \times n}$ é uma matriz triangular superior.

A decomposição QR é utilizada em problemas de otimização e regressão linear, sendo eficiente para resolver sistemas de equações lineares [Golub & Van Loan, 1996].

No contexto de aprendizado de máquina, a fatoração de matrizes é crucial para sistemas de recomendação. Considere uma matriz de interações $\mathbf{R} \in \mathbb{R}^{m \times n}$, onde m representa o número de usuários e n o número de itens. A fatoração de matrizes busca decompor $\mathbf{R}$ em duas matrizes menores:

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}^T \tag{2.13}$$

onde $\mathbf{P} \in \mathbb{R}^{m \times k}$ contém os fatores latentes dos usuários; $\mathbf{Q} \in \mathbb{R}^{n \times k}$ contém os fatores latentes dos itens; $k \ll \min(m,n)$ é a dimensão do espaço latente.

A predição da interação entre um usuário u e um item i é dada por:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i \tag{2.14}$$

onde $\mathbf{p}_u$ e $\mathbf{q}_i$ são os vetores latentes do usuário e do item, respectivamente. Métodos como Non-negative *Matrix Factorization* (NMF) e *Alternating Least Squares* (ALS) são amplamente utilizados para ajustar esses fatores [Koren et al., 2009].

A escolha da técnica de fatoração depende do problema e das propriedades desejadas na decomposição. Fatorizações como LU e Cholesky são úteis para sistemas lineares,

enquanto a SVD e a fatoração de matrizes não-negativas são ideais para aprendizado de máquina e processamento de sinais.

Outra abordagem fundamental é a *Low-rank Matrix Factorization*, onde assume-se que a matriz original pode ser aproximada por uma matriz de baixa complexidade, reduzindo significativamente o número de parâmetros necessários. Essa técnica é frequentemente utilizada em aprendizado de máquina para identificar estruturas latentes nos dados e melhorar a eficiência de armazenamento e processamento [Candès & Recht, 2009]. A fatoração de matrizes de baixa complexidade também desempenha um papel crucial na modelagem de dados esparsos, sendo amplamente aplicada em recomendações personalizadas e modelagem de dados ruidosos.

A *Low-rank Matrix Factorization* é especialmente eficaz na representação de grandes conjuntos de dados, pois permite a decomposição em componentes relevantes, eliminando redundâncias e minimizando o impacto do ruído. Seja uma matriz original $\mathbf{X} \in \mathbb{R}^{m \times n}$, sua aproximação de baixa complexidade pode ser expressa como:

$$\mathbf{X} \approx \mathbf{LR} \tag{2.15}$$

onde $\mathbf{L} \in \mathbb{R}^{m \times k}$ e $\mathbf{R} \in \mathbb{R}^{k \times n}$ com $k \ll \min(m,n)$ são matrizes de baixa complexidade que preservam as características principais da matriz original.

Diferentes métodos, como Singular Value Thresholding (SVT) e Robust PCA, são frequentemente utilizados para encontrar tais aproximações [Recht et al., 2010]. Essas técnicas são fundamentais para processamento de imagens e compressão de dados, permitindo a recuperação eficiente de estruturas latentes em grandes bases de dados.

2.4 Considerações Finais

Neste capítulo, apresentamos os principais fundamentos teóricos que sustentam o estudo dos modelos de Redes Neurais Profundas e seu comportamento durante o processo de aprendizado. Inicialmente, discutimos a estrutura básica das redes neurais artificiais, enfatizando o papel dos neurônios, dos pesos e das funções de ativação na formação dos padrões e na aprendizagem dos dados. A partir dessa base, aprofundamos a discussão em redes neurais profundas, demonstrando como a composição de múltiplas camadas possibilita a extração hierárquica de características, desde elementos simples até representações complexas, essenciais para a obtenção de embeddings discriminativos.

Além disso, abordamos o paradigma do aprendizado contínuo, destacando a dificuldade imposta pelo fenômeno do esquecimento catastrófico quando o modelo é treinado sequencialmente em diferentes tarefas. Foram exploradas diversas estratégias para mitigar esse

problema, como a regularização dos parâmetros (por meio do EWC), técnicas de replay e a expansão dinâmica da arquitetura. Essas abordagens ilustram a necessidade de preservar o conhecimento previamente adquirido enquanto se incorpora novas informações, estabelecendo um delicado equilíbrio entre plasticidade e estabilidade.

Por fim, apresentamos a fatoração de matrizes como uma ferramenta poderosa para a redução de dimensionalidade e extração de estruturas latentes dos dados, reforçando sua aplicação em problemas de recomendação e processamento de sinais.

Com esses fundamentos, estabelecemos as bases necessárias para compreender os desafios enfrentados durante o treinamento de redes neurais em contextos dinâmicos. No próximo capítulo, aprofundaremos a discussão sobre o problema do esquecimento catastrófico, investigando seus mecanismos subjacentes e apresentando estratégias avançadas para sua mitigação.

3

Esquecimento Catastrófico

O esquecimento catastrófico é um fenômeno que ocorre nos modelos de inteligência artificial que tem conhecimento para resolver uma tarefa e é submetido a aprender uma nova. Após aprender a nova tarefa, como o conhecimento antigo não teve um reforço, a assertividade do modelo degrada consideravelmente na tarefa original [McCloskey & Cohen, 1989].

As redes neurais são os modelos que mais sofrem esse problema. Isso porque elas armazenam os padrões aprendidos em seus pesos e esses pesos mudam a cada sessão de treinamento para capturar os padrões das amostras mais recentes [Robins, 1993, 1995].

Existem diversos métodos que já foram propostos para mitigar o problema do esquecimento catastrófico em RNP, embora ainda seja um problema em aberto. Para organizar e facilitar a compreensão dessas abordagens, Aleixo et al. [2024] propuseram uma taxonomia que classifica os principais modelos e estratégias, conforme ilustrado na Figura 3.1.

Nesta taxonomia, os métodos são agrupados em quatro categorias principais: i) *rehearsal*, ii) abordagens baseadas em distância, iii) sub-redes e iv) redes dinâmicas. Além dessas categorias, há também a classificação de modelos híbridos, que combinam características de mais de um grupo. Essas técnicas podem ser aplicadas em diferentes paradigmas de aprendizado, incluindo aprendizado supervisionado, não supervisionado e aprendizado por reforço.

3.1 Taxonomia

Nós classificamos um método como *rehearsal* quando ele utiliza alguma técnica para reforçar conhecimentos adquiridos anteriormente ao aprender novas tarefas. Alguns métodos armazenam parte dos dados, outras criam mecanismos para simular os dados. Os métodos baseados em distância utilizam as distância entre as amostras para agrupa-las, assim, uma amostra similar com outra é classificada como sendo da mesma classe. Sub-redes são

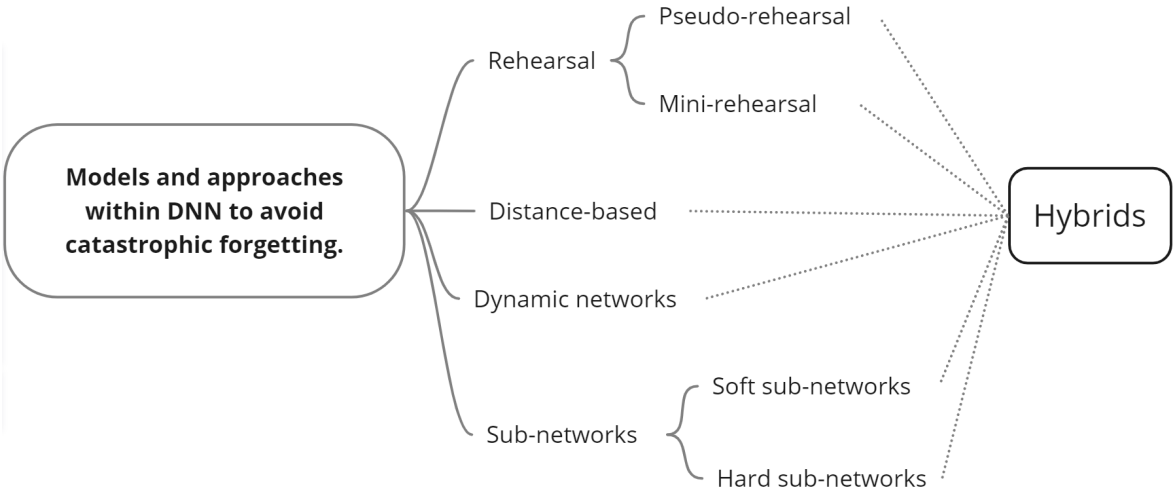


Figura 3.1. Taxonomia de abordagens para evitar o esquecimento catastrófico em Redes Neurais Profundas. Os métodos são categorizados em quatro grupos principais: *rehearsal*, abordagens baseadas em distância, sub-redes e redes dinâmicas. Além disso, alguns modelos apresentam características de múltiplos grupos, sendo classificados como métodos híbridos. Figura extraída do *survey* de Aleixo et al. [2024].

métodos que buscam selecionar uma parte do modelo para cada tarefa, minimizando a sobreposição dos parâmetros aprendidos. Alguns autores tentam encontrar a solução de todas as tarefas a serem aprendidas em um local próximo no hiperespaço dos parâmetros, enquanto outros fixam uma sub-parte para resolver a tarefa do momento. Por fim, classificamos as técnicas que alteram a estrutura das RNP (número de nós e camadas) como redes dinâmicas.

3.1.1 Rehearsal

Rehearsal é o processo de repetição de uma tarefa previamente aprendida enquanto uma nova é incorporada, com o objetivo de evitar o esquecimento. Essa técnica é amplamente utilizada na psicologia humana e foi introduzida no estudo de redes neurais por Robins Robins [1993, 1995]. No entanto, armazenar todos os dados que um modelo recebe ao longo de sua vida útil e utilizá-los para retreinamento pode ser impraticável, pois exige um grande volume de armazenamento e pode aumentar significativamente o tempo necessário para o treinamento.

As de três principais razões para que os métodos que utilizam técnicas de *rehearsal* sofram do problema de esquecimento são: i) magnitudes desequilibradas; ii) deriva semântica; e, iii) ambiguidades. [Belouadah & Popescu, 2020, Hou et al., 2019, Wu et al., 2019]

Para mitigar esses desafios, alguns pesquisadores têm se concentrado na redução da quantidade de dados que precisam ser armazenados, enquanto outros buscam otimizar o tempo de treinamento [Borsos et al., 2020, Hu et al., 2019, Leontev et al., 2019, Sprechmann

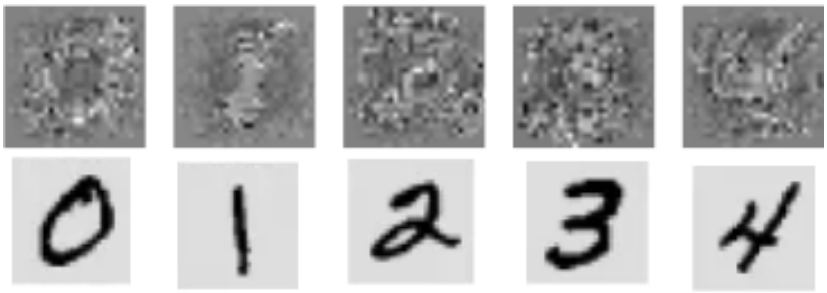


Figura 3.2. A primeira linha contém imagens ruidosas que representam as classes de 0 a 4 em uma rede neural treinada por Mellado et al. [2017]. A segunda linha contém imagens reais das mesmas classes extraídas do conjunto MNIST. Figura extraída do *survey* de Aleixo et al. [2024].

et al., 2018]. Essas abordagens são conhecidas como *mini-rehearsal* e *pseudo-rehearsal*, e serão discutidas a seguir.

3.1.1.1 Pseudo-rehearsal

Pseudo-*rehearsal* baseia-se na suposição de que armazenar dados previamente vistos não é essencial para lidar com o esquecimento catastrófico (CF). Em vez disso, é suficiente gerar novos dados sinteticamente, conhecidos como *pseudo-samples*, para representar classes previamente aprendidas sob demanda. Como ilustrado na Figura 3.2, os dados gerados não precisam ser idênticos aos originais.

Na figura, a primeira linha exibe imagens ruidosas geradas por Mellado et al. [2017] para representar classes do conjunto de dados MNIST [LeCun et al., 1998], especificamente os dígitos 0, 1, 2, 3 e 4. Embora os exemplos da primeira linha não sejam reconhecíveis como números por um humano, eles são projetados para ativar as mesmas conexões neuronais da rede original, que utiliza as imagens reais exibidas na segunda linha. Dessa forma, podem ser utilizados para reentreinar o modelo periodicamente, garantindo que conceitos previamente aprendidos sejam preservados.

Para iniciar uma nova sessão de treinamento e adquirir conhecimento adicional, uma cópia do modelo existente é criada, denominada *clone* do modelo. Em seguida, algumas imagens ruidosas geradas sinteticamente são rotuladas por esse modelo clonado. Os pares de imagens ruidosas e suas respectivas etiquetas, que apresentam alta confiança na predição do modelo clonado, são incorporados ao conjunto de dados atual como dados sintéticos. Assim, o modelo original é treinado nesse conjunto expandido, permitindo que ele retenha informações sobre tarefas passadas.

Entretanto, um problema central no uso de amostras sintéticas é que elas podem não pertencer à classe para a qual foram atribuídas. Esse fenômeno é conhecido como *problema de ambiguidades* no *Incremental Learning* [Masana et al., 2020], podendo levar a confli-

tos com as novas classes que o modelo precisa aprender. Como exemplo, considere uma rede neural treinada para classificar imagens de gatos e cachorros. Se uma nova classe for adicionada, como pássaros, e o algoritmo gerar uma imagem aleatória para essa classe, o modelo pode classificá-la erroneamente como um gato ou cachorro, pois só conhece essas duas categorias.

Técnicas como *Bayesian Learning* e *Stochastic Langevin Dynamics* foram empregadas por Leontev et al. [2019] para aprimorar a geração desses dados sintéticos. Em vez de criar amostras aleatoriamente, essas abordagens maximizam a ativação neuronal das classes desejadas. No entanto, tais técnicas podem resultar em dados sintéticos com baixa diversidade. Para mitigar esse problema, os autores propuseram o uso de um modelo baseado no movimento *browniano* [Karatzas & Shreve, 1991], permitindo evitar a seleção de amostras redundantes dentro do espaço de entrada e garantindo maior diversidade no conjunto de dados sintéticos.

Essa técnica busca gerar dados sintéticos que ativem neurônios específicos associados a determinadas classes. No entanto, não há garantia de que as ativações mais relevantes serão geradas. Para solucionar esse problema, Smith et al. [2021] propuseram o uso de um modelo invertido treinado na tarefa anterior para gerar *pseudo-samples* utilizando o rótulo da classe como entrada. No entanto, as amostras geradas ainda não são totalmente fiéis às originais. Posteriormente, PourKeshavarzi et al. [2022] aplicaram essa técnica com uma abordagem diferente, demonstrando que o uso de *Knowledge Distillation* (KD) [Hinton et al., 2015] melhora significativamente os resultados. Eles denominaram essa estratégia como *memory recovery paradigm*. Mais recentemente, Liu et al. [2022] utilizaram o modelo *DeepInversion* [Yin et al., 2020] para gerar *pseudo-samples* no contexto de *Few-shot Incremental Learning*.

Redes generativas também foram adotadas para criar amostras sintéticas mais semelhantes às originais. Por exemplo, Mellado et al. [2017] utilizaram uma *Recurrent Neural Network* (RNN) chamada DRAW [Gregor et al., 2015] para esse fim. Diferente das RNNs convencionais, esse modelo possui duas cabeças: (i) uma rede totalmente conectada com ativação *softmax* para classificar a entrada e (ii) um decodificador responsável por reconstruir a entrada original. Esse estudo demonstrou que as *Generative Adversarial Networks* (GANs) [Goodfellow et al., 2014b] podem desempenhar um papel crucial na mitigação dos efeitos do *Catastrophic Forgetting* (CF) no contexto do *pseudo-rehearsal*. No mesmo ano, Shin et al. [2017] propuseram o *Deep Generative Replay*, um modelo no qual uma GAN é utilizada como modelo auxiliar no início de cada sessão de treinamento para gerar amostras sintéticas de classes já conhecidas. A GAN é treinada ao final de cada sessão utilizando tanto o conjunto de dados da tarefa atual quanto os dados gerados por ela mesma.

GANs também foram empregadas para evitar CF em cenários mais gerais, como: (i)

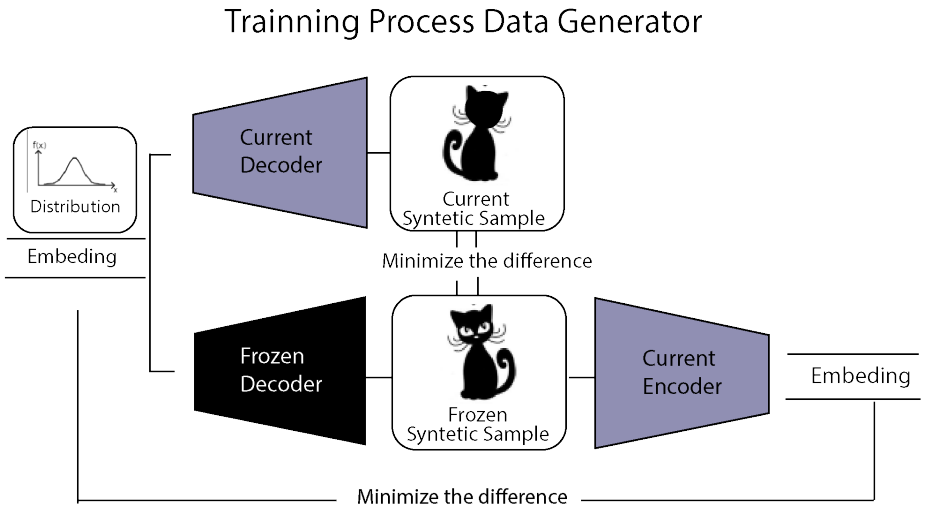


Figura 3.3. Processo de treinamento no *framework* PGMA. O *Encoder* gera uma representação da instância da imagem, que é usada pelo DPG para criar um conjunto de parâmetros para adaptar o *Solver*. O *Solver* então classifica a instância da imagem. Figura extraída do *survey* de Aleixo et al. [2024].

geração de imagens condicionadas a rótulos, onde a entrada é um rótulo de classe e a saída é uma imagem correspondente; e (ii) geração de imagens condicionadas por outra imagem, onde a entrada é uma imagem de uma classe e a saída é uma versão alternativa da mesma classe em um domínio diferente. Por exemplo, uma imagem de um gato no estilo de desenho animado pode ser transformada em um esboço em preto e branco [Zhai et al., 2019]. No entanto, os dados gerados por modelos generativos nem sempre apresentam alta qualidade, frequentemente exibindo artefatos como *blur*. Para superar essa limitação, Xiang et al. [2019] propuseram a geração de *feature maps*, que são inseridos nas camadas internas do classificador em vez de criar diretamente amostras sintéticas na entrada da rede. Essa abordagem facilita a classificação, pois os *feature maps* possuem menor dimensionalidade e capturam padrões relevantes para o modelo, reduzindo a complexidade da tarefa em comparação com o uso de imagens brutas.

O método *Learning using Encoded Experience Replay* (CLEER) [Rostami et al., 2019] utiliza um *Autoencoder* (AE) treinado para garantir que todas as tarefas compartilhem a mesma distribuição no espaço de representação. Para isso, o modelo é dividido em três componentes principais: (i) um *encoder*; (ii) um classificador compartilhado; e (iii) um *decoder* responsável por gerar *pseudo-samples*. O *encoder* é treinado para gerar representações com a mesma distribuição, independentemente da tarefa. Isso é alcançado modelando o espaço de representação como uma distribuição multimodal utilizando o *Gaussian Mixture Model* (GMM). Quando uma nova tarefa precisa ser aprendida, o modelo utiliza o *decoder* junta-

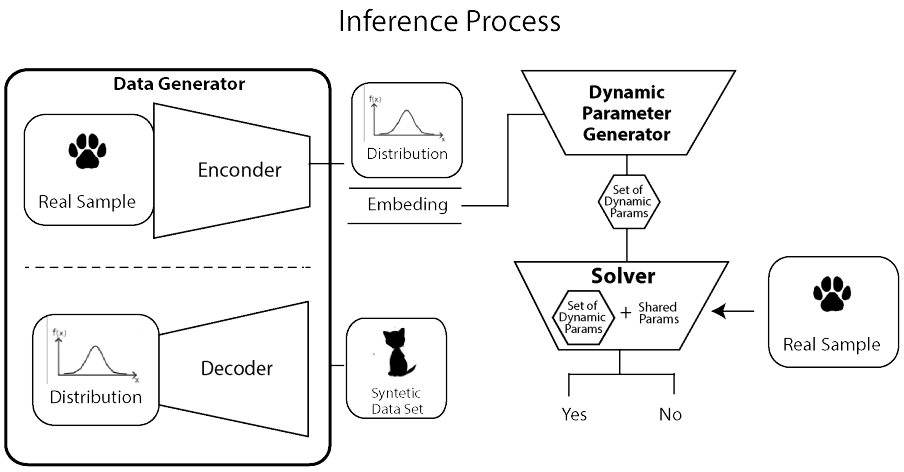


Figura 3.4. Processo de inferência no *framework* PGMA. O *Decoder* gera um conjunto de dados sintético para evitar CF ao aprender novas classes, garantindo a continuidade do aprendizado sem esquecer tarefas anteriores.

mente com o GMM para gerar *pseudo-samples*, que são então mesclados com as amostras da tarefa atual. Com esse conjunto de dados expandido, o modelo passa por uma nova fase de treinamento em seus três componentes. Esse processo incentiva o *encoder* a agrupar todas as amostras em uma única região do espaço de representação, simplificando a tarefa de categorização e funcionando como um mecanismo de normalização.

O *Variational Autoencoder* (VAE) é naturalmente mais resistente ao *Catastrophic Forgetting* (CF) [van de Ven et al., 2020]. Por essa razão, o VAE tem sido utilizado em *Incremental Learning* tanto para a geração de imagens quanto para tarefas de classificação. A escolha da distribuição *prior* no espaço latente é um fator crítico para que um VAE obtenha resultados precisos [Egorov et al., 2021]. No entanto, no contexto de *Incremental Learning*, a definição dessa distribuição é um desafio, pois os dados evoluem continuamente ao longo do tempo. Egorov et al. [2021] propuseram o uso do gradiente em um treinamento *end-to-end* para encontrar uma distribuição *prior* que seja uma combinação da distribuição dos dados atuais e das distribuições anteriores. No entanto, essa abordagem também apresenta problemas de qualidade nas imagens geradas pelo *decoder*.

Outra estratégia utilizada para representar classes previamente aprendidas é o uso de *prototypes*. O modelo pode lembrar uma classe por meio de um protótipo, que atua como um ponto de referência. Um protótipo é uma representação média das amostras de uma determinada classe, calculada a partir da média de todas as amostras disponíveis. Essa abordagem requer menos armazenamento de memória, mas exige que todas as amostras de uma classe estejam agrupadas e bem separadas das demais classes. Para mitigar essa limitação, é possível adicionar ruído Gaussiano ao protótipo, gerando uma coleção efêmera de amostras que

podem ser utilizadas no conjunto de treinamento de cada sessão de aprendizado [Petit et al., 2023, Zhu et al., 2021b].

Em vez de forçar a aproximação das amostras no espaço de representação, Hu et al. [2019] propõem a adaptação dos pesos do modelo para lidar com instâncias de diferentes classes. Para isso, o modelo é empregado juntamente com três modelos auxiliares: (i) um *Solver*, que prevê a resposta de uma tarefa; (ii) um *Dynamic Parameter Generator* (DPG), que gera um conjunto de parâmetros a ser incorporado ao *Solver* para cada amostra; (iii) um *Encoder*, responsável por criar uma representação da entrada que será utilizada pelo DPG e pelo *Decoder*; e (iv) um *Decoder*, que gera amostras sintéticas realistas de tarefas que não estão mais disponíveis. Essa estratégia permite a mutação dos pesos do modelo com base na amostra processada no momento, ajudando a mitigar o problema de *semantic drift* [Masana et al., 2020]. Esse método é classificado como *pseudo-rehearsal* (e não uma *Dynamic Network*), pois o modelo mantém sua estrutura original, apenas modificando os valores de seus parâmetros ao utilizar dados derivados do conhecimento prévio.

A Figura 3.3 e 3.4 ilustram como esses modelos interagem durante as fases de inferência e treinamento, respectivamente. Dada uma imagem de uma nova classe, o *Encoder* gera um vetor de representação. Esse vetor é então passado para o DPG, que produz um conjunto de pesos. Esses pesos são combinados com aqueles fornecidos pelo *Solver* para prever a classe da entrada y, juntamente com as amostras da nova tarefa. O modelo é treinado utilizando amostras geradas pelo *Decoder*.

O *Data Generator*, um componente macro composto pelo *Encoder* e pelo *Decoder*, também pode sofrer com o problema de *Catastrophic Forgetting* (CF). Para mitigar esse efeito, os autores propõem o uso de uma função de perda baseada em *Knowledge Distillation* (KD), conforme ilustrado na Figura 3.4. Uma versão congelada do modelo da última sessão de treinamento é utilizada como "professor"para a sessão atual, preservando o comportamento aprendido anteriormente. Para isso, um vetor de representação aleatório, gerado a partir de uma distribuição normal, é utilizado para criar uma imagem sintética. Em seguida, o objetivo é minimizar a diferença entre essa imagem sintética e aquela gerada pelo *Decoder* atual. Além disso, o *Encoder* deve gerar uma representação da imagem sintética e minimizar a diferença entre essa nova representação e a usada pelo *Decoder* congelado.

Embora os resultados indiquem um bom desempenho ao longo de uma sequência de duas tarefas, avaliações adicionais são necessárias para validar a eficácia do método em lidar com sequências arbitrárias de tarefas e cenários mais complexos além do MNIST. Investigações futuras devem focar na ampliação da avaliação para contemplar diferentes contextos, garantindo a escalabilidade e robustez da abordagem. No entanto, o uso de conjuntos de dados sintéticos mais semelhantes às amostras reais demonstra ser uma abordagem promissora.

Hypernetworks também têm sido empregadas para gerar os valores dos parâmetros do

modelo com base em um identificador de tarefa [von Oswald et al., 2020]. Essa abordagem transfere o problema de CF para a *hypernetwork*. No entanto, von Oswald et al. [2020] argumentam que esse efeito é reduzido, pois as relações entre pares de entrada e saída são menos complexas do que as relações entre os dados de diferentes classes ao longo das tarefas. Ainda assim, o problema persiste. Para lidar com isso, os autores utilizam um *Variational Autoencoder* (VAE) para aplicar *pseudo-rehearsal* dentro da *hypernetwork*.

Hu et al. [2021] explicam o problema de *Catastrophic Forgetting* (CF) como um efeito causal da perda de dados antigos durante o treinamento com novos dados. Eles argumentam que simplesmente repetir representações de características dos dados não captura as informações causais contidas na reprodução dos dados originais. No entanto, os autores demonstraram que é possível destilar o efeito de colisão entre os dados antigos e novos no espaço de representações (*embedding space*).

Outros algoritmos de aprendizado também foram testados no contexto de *pseudo-rehearsal*. Por exemplo, *Contrastive Hebbian Learning* [Movellan, 1991] e *No-Prop* [Widrow et al., 2013] foram avaliados por Hattori & Tsuboi [2018]. No entanto, esses estudos foram limitados a conjuntos de dados simplificados (*toy datasets*) e redes rasas (*shallow networks*). Pesquisas futuras podem explorar a aplicação dessas abordagens em arquiteturas profundas e conjuntos de dados mais complexos, além de investigar o uso de GANs para gerar dados sintéticos. Contudo, neste trabalho, focamos em métodos baseados no algoritmo de aprendizado por descida de gradiente, tornando essas abordagens fora do nosso escopo.

A Figura 3.5 apresenta uma visão geral dessa categoria. Ela ilustra como os *pseudo-exemplos* podem ser utilizados tanto para modificar os valores dos parâmetros do modelo quanto para destilar informações causais. Além disso, a figura destaca as principais estratégias para a geração de *pseudo-samples*. Na próxima seção, apresentamos abordagens que permitem o armazenamento de certos dados antigos como uma estratégia para prevenir o CF.

3.1.1.2 Mini-rehearsal

Mini-*rehearsal* utiliza um subconjunto dos dados originais deve ser mantido para evitar o esquecimento catastrófico (CF) durante o treinamento de um modelo em uma nova tarefa. O objetivo principal é minimizar a quantidade de dados antigos que precisam ser armazenados, permitindo que o modelo aprenda a nova tarefa sem sofrer com CF. Isso pode ser alcançado por meio de uma amostragem seletiva de dados. Esse subconjunto é conhecido como *coreset*, e sua principal função é representar de forma adequada o conjunto de treinamento original. Assim, com esse subconjunto armazenado, o CF pode ser controlado pelo modelo.

O *Greedy Sampler and Dumb Learner* (GDumb), um *baseline* consolidado dessa categoria, demonstra a força dessa abordagem. Trata-se de um modelo ingênuo que questiona os

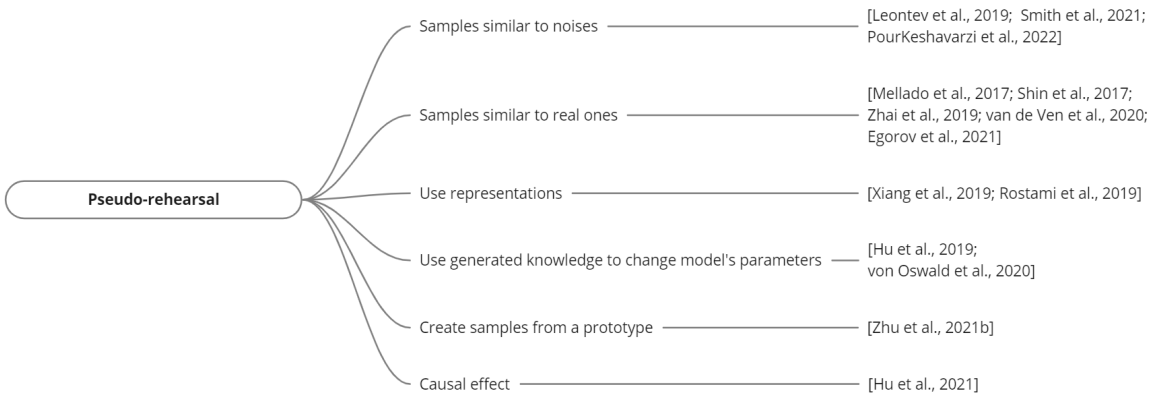


Figura 3.5. Visão geral das principais abordagens na categoria *pseudo-rehearsal*. Em vez de armazenar dados reais, o modelo gera amostras sintéticas conforme necessário. Figura extraída do *survey* de Aleixo et al. [2024].

avanços das abordagens contemporâneas para mitigar o CF sugeridas na literatura [Prabhu et al., 2020]. O GDumb é composto por dois módulos principais: um gerenciador de memória e um aprendiz. Para cada nova classe aprendida, um novo *bucket* é criado para armazenar suas amostras, seguindo uma estratégia de gerenciamento ingênua. Quando a memória atinge sua capacidade máxima, a amostra do *bucket* com maior número de amostras é removida. O aprendiz, por sua vez, utiliza as amostras armazenadas na memória para treinar uma rede neural profunda (*DNN*) do zero. Esse modelo supera diversas abordagens apresentadas na literatura.

Há uma grande quantidade de pesquisas voltadas para aprimorar a seleção de amostras no *coreset* [Borsos et al., 2020, Hayes et al., 2019, Liu et al., 2021, Shankar & Sarawagi, 2018, Yoon et al., 2022]. Uma estratégia promissora envolve a utilização de uma otimização bilevel baseada em gradientes para selecionar o *coreset* de modo a minimizar o esquecimento de conhecimento ao longo do treinamento [Borsos et al., 2020]. No entanto, essa abordagem possui um alto custo computacional, mesmo com as otimizações propostas pelos autores. Por exemplo, ao utilizar uma GPU GeForce GTX 1080 Ti, a seleção de um *coreset* com 100 amostras em um conjunto de 1000 leva aproximadamente 20 segundos, enquanto a seleção de 400 amostras no mesmo conjunto leva mais de três minutos. Isso demonstra que o tempo de processamento não cresce de forma linear. Além disso, essa abordagem funciona apenas com um conjunto de amostras estático para cada tarefa, tornando-a inviável para cenários de *Online Incremental Learning*.

A maioria dos pesquisadores defende que o tamanho do *coreset* deve ser limitado, mas há duas formas de abordar essa questão: (i) manter um número fixo de amostras por classe, denotado por M_c; ou (ii) gerenciar um total de M amostras distribuídas entre todas as classes. A primeira abordagem apresenta o problema do crescimento exponencial da memória

conforme novas tarefas são aprendidas. Para mitigar essa questão, alguns pesquisadores propõem a compressão dos dados como uma solução viável [Hayes et al., 2019, Wang et al., 2022a].

O algoritmo *Exemplar Streaming* (ExStream) realiza a compressão dos dados armazenados nos *buckets* e foi comparado com outros dois algoritmos de compressão: (i) *Online k-means* e (ii) *CluStream*; além de dois algoritmos de substituição de amostras: (i) *Reservoir Sampling* e (ii) *First-In-First-Out* (FIFO) queue. Os resultados mostraram que, quando o tamanho do *bucket* é maior que 256 amostras, todos os algoritmos apresentaram desempenhos semelhantes. No entanto, quando o tamanho do *bucket* é pequeno, como quatro ou oito amostras, a escolha adequada das amostras pode impactar significativamente os resultados. Por exemplo, o algoritmo ExStream demonstrou um aumento de mais de 50% no desempenho em alguns cenários, quando comparado ao *reservoir sampling* [Hayes et al., 2019].

A compressão das amostras gera um compromisso entre qualidade e quantidade, no entanto, é possível obter ganhos em acurácia mesmo com amostras de menor qualidade [Wang et al., 2022a]. A literatura também aponta o uso de mutações nas amostras do *coreset* como uma forma de maximizar a retenção de memória. O método *Mnemonics* [Liu et al., 2020c] considera cada *bucket* como uma variável e aplica uma fase de pós-treinamento, na qual as amostras armazenadas na memória sofrem mutações para se tornarem mais representativas. Apesar de ser mais eficiente computacionalmente do que os métodos mencionados anteriormente, esse processo ainda precisa ser realizado na fase *offline*. Já Jin et al. [2021] propuseram a edição das amostras do *coreset* com base no gradiente, de forma *online*, em vez de substituí-las. Entretanto, seu estudo focou em um cenário de aprendizado incremental sem fronteiras, onde a distribuição dos dados muda continuamente sem uma identificação explícita das tarefas.

Outra abordagem para reduzir a quantidade de memória necessária é armazenar representações (*feature maps*) em vez das amostras brutas para realizar o processo de *rehearsal*. O método *REMIND* [Hayes et al., 2020] mostrou que essa estratégia não apresenta melhores resultados em cenários onde cada tarefa é aprendida sequencialmente. No entanto, em cenários onde as classes são apresentadas em qualquer ordem (*online learning*), esse método obteve resultados de estado da arte. Para alcançar esse desempenho, os autores utilizaram a técnica *MixUp* [Liang et al., 2018] nas amostras armazenadas, tornando o processo de *replay* mais robusto e representativo.

Dado que o *coreset* possui um tamanho fixo de M, é necessário um método para substituir, atualizar ou remover amostras dos *buckets* das classes quando a memória estiver cheia. O algoritmo de seleção *herding*, introduzido pelo *Incremental Classifier and Representation Learning* (ICaRL) [Rebuffi et al., 2017], descrito na Seção 3.1.5 como uma técnica híbrida, foi um dos pioneiros nesse campo. O *herding selection* escolhe as amostras mais próximas

ao centróide das amostras de cada classe. Já o *reservoir sampling* é a estratégia mais utilizada para gerenciar o *coreset*. Essa abordagem ingênua oferece a cada amostra a mesma probabilidade de ser selecionada para remoção quando o *coreset* atinge sua capacidade máxima. Como resultado, não se pode garantir que o *coreset* permanecerá sempre balanceado, embora estudos empíricos indiquem que essa estratégia proporciona uma distribuição aproximada [Hayes et al., 2019].

Diversas métricas podem ser empregadas para definir a seleção das amostras que serão mantidas no *coreset*. O *Shapley value* foi utilizado para estimar a contribuição individual de cada amostra no desempenho do modelo [Shim et al., 2021]. O método *Maximally Interfered Retrieval* (MIR) utiliza a perda para verificar se a amostra recém-chegada sofre um impacto negativo maior na atualização dos parâmetros do modelo do que as amostras já armazenadas. Além disso, a teoria da informação foi aplicada para gerenciar as amostras mais relevantes a serem mantidas no *coreset*, utilizando um modelo Bayesiano para calcular os critérios de forma eficiente, explorando estruturas de matrizes de posto um [Sun et al., 2022].

Outra métrica bastante explorada na literatura é a confiança do classificador. É possível selecionar as amostras com maior ou menor nível de confiança. No primeiro caso, são escolhidas amostras que o modelo consegue classificar com alta precisão; no segundo, são escolhidas amostras em que o modelo apresenta maior dificuldade de inferência. De acordo com estudos empíricos, selecionar as amostras de maior confiabilidade resulta em um melhor desempenho geral [He et al., 2018].

Um agente de *Reinforcement Learning* (RL) foi treinado para decidir qual proporção da memória deve ser alocada para cada classe e quais amostras devem ser mantidas [Liu et al., 2021]. Como o agente também pode sofrer com *Catastrophic Forgetting* (CF), ele é treinado apenas na primeira tarefa, dividindo os dados em N grupos para simular várias tarefas. Em um cenário de *Online Incremental Learning*, o problema de conjuntos de dados desbalanceados e ruidosos torna-se ainda mais predominante. O método *Online Coreset Selection* [Yoon et al., 2022] gerencia a memória seguindo três objetivos principais: (i) *minibatch similarity*, para selecionar amostras representativas da tarefa atual; (ii) *cross-batch diversity*, para reduzir a redundância entre as amostras da tarefa atual; e (iii) *coreset affinity*, para minimizar a interferência entre as amostras selecionadas e o conhecimento das tarefas anteriores.

Todos os modelos previamente treinados podem utilizar o *coreset* em um cenário de *Incremental Learning*, transformando-o em uma *Memory Augmented Neural Network* (MANN) [Santoro et al., 2016]. A memória da MANN é representada como uma tabela em que cada linha é uma tupla $\{l_m, v_m, \alpha_m\}$, onde l_m é o rótulo da instância, v_m é a representação (*embedding*) gerada pela última camada do modelo (antes da ativação *softmax*) e α_m é o peso associado à linha. Durante a inferência, a previsão do modelo pré-treinado é interpolada com

a previsão da memória, utilizando as N tuplas mais similares, comparando v_m com a representação gerada pela última camada do modelo. Esses dois valores são combinados por meio de um conjunto de pesos, gerado dinamicamente por uma *Recurrent Neural Network* (RNN) a cada previsão.

Durante o treinamento, o rótulo verdadeiro é utilizado para atualizar a memória caso o modelo cometa um erro. Além disso, apenas as tuplas em que l_m corresponde ao rótulo verdadeiro são atualizadas, garantindo que classes raras não sejam esquecidas. A RNN também é treinada de forma *online* para determinar a relevância da previsão feita pela memória e pelo modelo pré-treinado. Embora essa abordagem sugira que o CF seja evitado, os autores não realizam avaliações específicas para verificar essa hipótese. Em vez disso, a avaliação se concentra na capacidade de *forward transfer learning*, sem considerar se as classes aprendidas anteriormente ainda são lembradas pelo modelo. No entanto, é evidente que essa memória auxilia na mitigação do problema de conjuntos de dados desbalanceados, explorando as N tuplas mais similares [Shankar & Sarawagi, 2018].

Seguindo a abordagem baseada em *feature embeddings* (saída da última camada), Sprechmann et al. [2018] propõem a utilização de uma *hash table*, onde a representação da entrada é usada como chave e o rótulo verdadeiro como valor. O modelo é dividido em três componentes principais: (i) um gerador de *embeddings*; (ii) a memória baseada em tabela de *hash*; e (iii) o classificador. Durante o treinamento, todos os módulos do modelo são atualizados. Na tabela de *hash*, as amostras são armazenadas em um *buffer* circular, enquanto o gerador de *embeddings* e o classificador são ajustados via *backpropagation*. Na inferência, o algoritmo *K-Nearest Neighbors* (KNN) é aplicado às chaves da tabela de *hash*, e o resultado é utilizado para modificar os pesos do classificador conforme necessário. Essa abordagem segue a ideia de adaptação *online* descrita por Hu et al. [2019], porém, ao invés de utilizar um modelo para gerar a mudança nos pesos, uma pequena sessão de treinamento é realizada para encontrar essa variação Δ. Como a memória possui um tamanho fixo e opera como um *buffer* circular, o modelo apresenta uma tendência a favorecer as classes mais recentemente treinadas, tornando difícil a retenção de classes raras ou tarefas antigas.

A literatura apresenta diversas abordagens para o uso do *coreset* na mitigação do *Catastrophic Forgetting* (CF). O método *MeRGAN* [Wu et al., 2018a] explora duas estratégias principais: (i) *joint training with replay* e (ii) *replay alignment*. Na primeira abordagem, as amostras são geradas condicionadas ao rótulo das tarefas anteriores, permitindo seu uso conjunto com as amostras da tarefa atual. Na segunda, aplica-se *Knowledge Distillation* (KD) no modelo gerador, tornando o gerador da tarefa atual um aluno do gerador aprendido na última tarefa.

O método *Gradient Episodic Memory* (GEM) [Lopez-Paz & Ranzato, 2017] utiliza o *coreset* para impor uma restrição rígida na etapa de atualização do modelo. O modelo só

aceita mudanças nos parâmetros se o erro sobre o *coreset* não aumentar. Isso evita o esquecimento das amostras do *coreset* e permite a transferência reversa de conhecimento. No entanto, o erro geral da classe ainda pode aumentar, e o modelo pode perder sua capacidade de aprender novas tarefas devido a essa restrição severa. Para contornar essa limitação, Chaudhry et al. [2019a] propuseram o método *A-GEM*, que relaxa essa restrição ao permitir atualizações mesmo que o erro aumente, desde que não ultrapasse um limite predefinido. Essa abordagem superou o *GEM* tanto em acurácia quanto no número de tarefas que o modelo consegue aprender.

Para regularizar o treinamento de maneira distinta, Tang et al. [2021] dividiram o gradiente em duas partes utilizando o *coreset*: (i) gradientes relativos a todas as tarefas; e (ii) gradientes específicos de cada tarefa. Em seguida, aplicaram restrições distintas para cada parte. No primeiro caso, os gradientes devem ter a mesma direção. No segundo, os gradientes específicos de cada tarefa devem ser ortogonais aos das demais tarefas. Outro método de regularização adotado foi o *Elastic Weight Consolidation* (EWC) [Kirkpatrick et al., 2017], utilizado no domínio de *Natural Language Generation* (NLG) [Mi et al., 2020a]. Essa técnica pertence à categoria de *Sub-networks*, que será discutida na Seção 3.1.3.

Hou et al. [2019] e Wu et al. [2019] levantaram três principais razões pelas quais métodos baseados em *mini-rehearsal* sofrem com o problema do esquecimento: (i) magnitudes desbalanceadas; (ii) *semantic drift*; e (iii) ambiguidades. Para mitigar esses problemas, Hou et al. [2019] desenvolveram o método *LUCIR*. Para lidar com o primeiro problema, os autores observaram que os valores dos *logits* referentes às classes da tarefa atual tendem a ser significativamente maiores do que os das classes aprendidas anteriormente. Esse fenômeno ocorre devido ao desbalanceamento de dados, uma vez que as amostras das classes da tarefa atual são abundantes, enquanto as amostras das classes antigas são escassas. Para corrigir esse desbalanceamento, aplicaram normalização cosenoidal aos *logits*, garantindo que todos possuam magnitudes similares. Para mitigar o *semantic drift*, utilizaram *Knowledge Distillation* (KD) na camada de *embedding*. Por fim, para lidar com ambiguidades entre classes, aplicaram uma função de perda baseada em *margin ranking*.

O método *BiC* adiciona uma camada final ao modelo, responsável por equalizar o viés dos *logits* em relação às novas classes [Wu et al., 2019]. Para isso, as amostras da tarefa atual são divididas em dois conjuntos: um para validação e outro para treinamento. O conjunto de treinamento, que ainda é abundante, é utilizado para treinar o modelo até a camada de *logits*, na etapa denominada *estágio 1* do treinamento. Em seguida, o conjunto de validação, que é balanceado em relação ao conjunto armazenado na memória, é utilizado para ajustar a última camada linear, responsável pelas previsões. Durante a primeira etapa, também é aplicado *Knowledge Distillation* (KD) para evitar o *semantic drift*. Os autores argumentam que, na segunda etapa, o fato de o modelo ser treinado em um conjunto de dados balanceado

permite que ele lide com o problema das ambiguidades de forma autônoma. O *BiC* não é categorizado como uma *Dynamic Network*, pois a camada adicional é incorporada apenas durante a criação do modelo, sem modificações ao longo de sua vida útil.

O problema das magnitudes desbalanceadas também foi observado por Belouadah & Popescu [2020]. Para lidar com essa questão, os autores utilizam memória extra para armazenar estatísticas dos nós do classificador referentes às classes da sessão de treinamento atual, quando os dados dessas classes são abundantes. Nas sessões de treinamento subsequentes, quando os dados dessas classes se tornam escassos, a magnitude de seus nós tende a diminuir. Assim, essas estatísticas são usadas para reescalar os pesos desses parâmetros. Além disso, Zhang et al. [2022] demonstraram que, em um cenário de *Online Incremental Learning*, aplicar transformações aleatórias sobre o *coreset* para aumentar seu tamanho auxilia o modelo na generalização do conhecimento e na mitigação do *Catastrophic Forgetting* (CF). Seu estudo fornece uma base sólida para esse tipo de configuração.

Lee et al. [2019] propõem o uso de um fluxo contínuo de dados não rotulados provenientes da internet para mitigar o problema de dados desbalanceados. Eles argumentam que essa abordagem reduz os três problemas mencionados anteriormente e que é mais eficaz do que *fine-tuning* do modelo em uma fase pós-treinamento com um conjunto de dados restrito. Os dados recebidos do fluxo contínuo são rotulados pelo próprio modelo, utilizando uma versão salva na última sessão de treinamento, permitindo destilar o conhecimento das classes antigas. No contexto de *object detection*, Dong et al. [2021] utilizam dados não rotulados para amostrar objetos que não estão mais presentes na tarefa atual. Além disso, empregam um esquema de redes duais para aplicar *Knowledge Distillation* (KD) e preservar o conhecimento aprendido anteriormente.

Ahn et al. [2021] argumentam que esse viés é causado principalmente pelo cálculo das probabilidades *softmax*, que combina os escores de saída para todas as classes antigas e novas. Como solução, propõem separar a camada *softmax*, criando uma *head* para cada tarefa e aplicando *Knowledge Distillation* utilizando o *coreset* durante o treinamento. No entanto, Zhu et al. [2021a] demonstram que os modelos também criam um viés no espaço de *embedding* para as classes mais recentes. Para lidar com isso, propõem a realização de uma *semantic augmentation* (*semanAug*) no espaço latente utilizando amostras antigas.

Belouadah et al. [2020] seguem essa linha de pesquisa e defendem que o esquecimento afeta principalmente a camada de classificação. Eles sugerem que o *fine-tuning* convencional pode mitigar o CF, pois os classificadores aprendidos em tarefas antigas podem ser mantidos para padronizar os pesos de todas as classes aprendidas. Os autores argumentam que isso é necessário devido ao viés em relação às novas classes. Assim, para tornar os pesos mais equilibrados após o aprendizado de uma nova classe, a padronização dos pesos iniciais é realizada.

Hou et al. [2018] propõem o método *Adaptation by Distillation* (AD) para demonstrar as vantagens da utilização de *Knowledge Distillation* no contexto de *Incremental Learning*. Durante o treinamento, o modelo principal não aprende diretamente a partir dos rótulos verdadeiros. Em vez disso, um modelo auxiliar, denominado "*expert*", é treinado com os dados originais. Após essa etapa, o modelo principal é treinado pelo *expert*, utilizando os rótulos suavizados (*soft labels*). Os autores demonstraram que esse processo fornece ao modelo um conhecimento mais generalizado, reduzindo os efeitos do CF. Apesar da ampla utilização do KD para mitigar o CF, Belouadah & Popescu [2019] mostram, em seus experimentos, que essa técnica pode ser prejudicial à acurácia do modelo quando este tem acesso a um *coreset* representativo.

A abordagem de *Meta-learning* foi proposta para criar um modelo agnóstico a tarefas [Rajasegaran et al., 2020]. Esse modelo não é especializado em nenhuma tarefa específica, mas sua solução está próxima de todas elas. Dessa forma, ele pode utilizar o *coreset* juntamente com a estratégia de *one-shot learning* para resolver qualquer tarefa ao longo de sua vida útil. Von Oswald et al. [2021] descobriram que o aprendizado esparso surge naturalmente nesse tipo de modelo (*MAML*), pois uma grande fração das taxas de aprendizado tende a se aproximar de zero. Eles demonstraram que isso pode ser explorado para direcionar o aprendizado para as regiões mais relevantes do modelo. Além disso, Perez-Rua et al. [2020] mostraram que essa abordagem também é eficaz no contexto de *object detection*. O método *XtarNet* [Yoon et al., 2020] utiliza *meta-learning* para treinar uma *MetaCNN*, que gera representações (*embeddings*) para novas classes. Essas novas representações são combinadas com aquelas geradas por uma rede previamente treinada, e o *embedding* resultante é utilizado para a classificação final. Parte dos dados de cada classe é mantida na memória para ser utilizada como conjunto de consulta (*query set*) e conjunto de suporte (*support set*) no *meta-learning*.

Classificadores generativos são menos propensos ao CF, pois os limites de decisão das *n* classes aprendidas sequencialmente permanecem os mesmos de quando são treinados em conjunto [Banayeeanzade et al., 2021]. O método *GeMCL* estende o *OML* [Javed & White, 2019], substituindo seu classificador discriminativo por um classificador generativo (*Bayesian classifier*). Além disso, Henning et al. [2021] propuseram uma estrutura Bayesiana onde distribuições de parâmetros condicionadas à tarefa são continuamente aprendidas e comprimidas em uma *Hypernetwork*. Eles relataram quase nenhum esquecimento e demonstraram que o método escala para arquiteturas modernas, como *ResNets*.

Recentemente, com o aumento do interesse em *Transformers*, pesquisadores começaram a investigar seus benefícios potenciais para lidar com o problema de CF [Wang et al., 2022b,c]. A hipótese proposta é que, ao identificar o *Prompt* apropriado, o modelo pode evitar CF de maneira eficaz. No entanto, um desafio surge na determinação do *Prompt* ideal para

cada instância. No estudo de Wang et al. [2022b], é apresentado um método para aprender e armazenar os *Prompts* associados a cada classe dentro do *coreset*. Cada *Prompt* é vinculado a uma chave, que corresponde ao *embedding* de uma amostra da respectiva classe. Durante a inferência, o *embedding* da amostra atual é utilizado para selecionar o *Prompt* mais adequado. Essa abordagem abre uma nova linha de pesquisa, pois desloca o foco da simples prevenção do CF para a descoberta do *Prompt* ideal para cada caso.

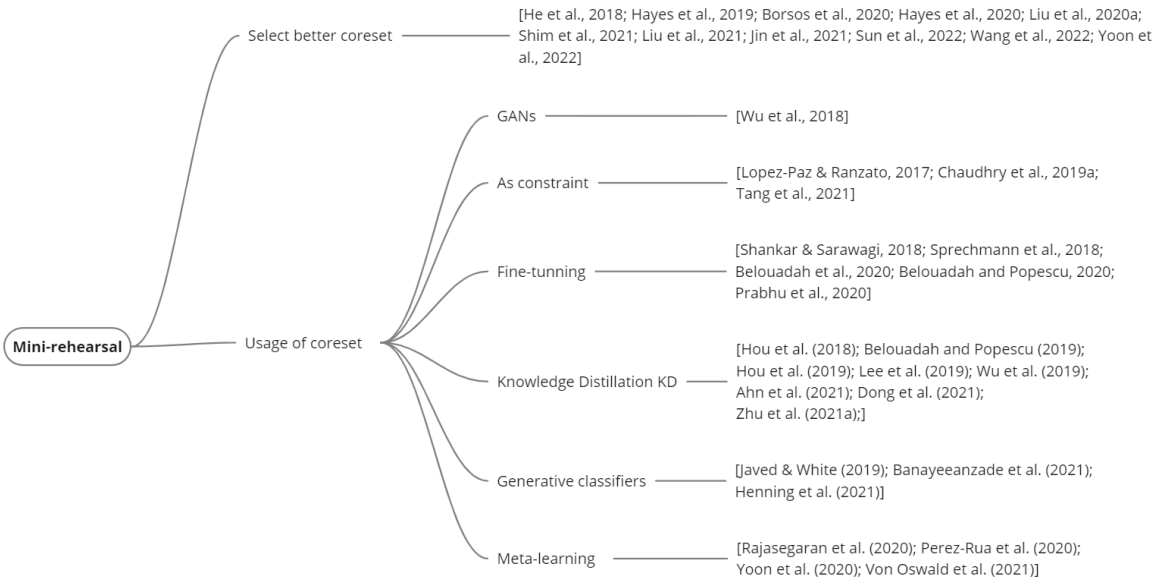


Figura 3.6. Visão geral das principais abordagens na categoria de *Mini-rehearsal*. No *Mini-rehearsal*, o modelo pode armazenar uma parte limitada das amostras de tarefas antigas. Existem trabalhos que utilizam esses dados para evitar CF, enquanto outros exploram formas mais eficientes de selecionar o *coreset*. Figura extraída do *survey* de Aleixo et al. [2024].

A Figura 3.6 apresenta uma visão geral dessa categoria. Embora os métodos de *rehearsal* apresentem bons resultados, eles sofrem com um problema inerente de *overfitting* no *coreset* [Chaudhry et al., 2019b]. Verwimp et al. [2021] argumentam que esse fenômeno ocorre porque os modelos que utilizam *coreset* permanecem na mesma região de baixa perda após o término de uma tarefa. No entanto, apontam que mais pesquisas nessa direção são necessárias.

Ainda há muitas lacunas na pesquisa para avaliar a melhor abordagem para selecionar as amostras mais adequadas a serem mantidas na memória. Na próxima seção, apresentamos trabalhos que utilizam técnicas focadas no aprendizado de representações, as quais classificamos como métodos *Distance-based*.

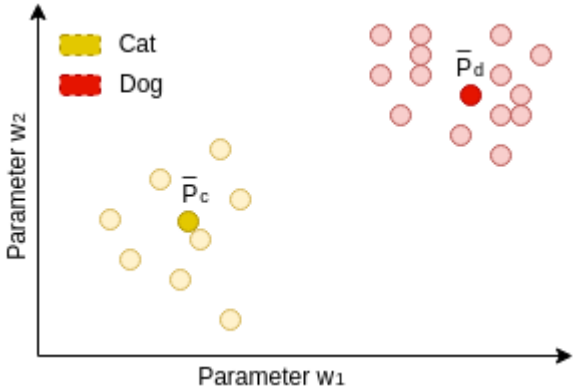


Figura 3.7. Uma representação prototípica que condensa toda a classe em um único ponto, reduzindo o impacto do desbalanceamento de dados entre as classes. Os pontos $\overline{p}_c$ e $\overline{p}_d$ servem como representações prototípicas dos pontos claros. Figura extraída do *survey* de Aleixo et al. [2024].

3.1.2 Distance-based

Distance-based é uma categoria suportada pela semelhança entre amostras da mesma classe no espaço de representações (*embeddings*). Rebuffi et al. [2017] propõem duas abordagens principais: (i) representação de dados fixa e (ii) aprendizado de representação. Na primeira abordagem, um gerador de *embedding* fixo é utilizado em cada etapa do *Incremental Learning*. O principal desafio é garantir que os *embeddings* gerados sejam representativos de todas as classes futuras, de forma que as amostras pertencentes à mesma classe permaneçam agrupadas, ao mesmo tempo em que são mantidas separadas das amostras de outras classes. Na segunda abordagem, o gerador de *embedding* também sofre ajustes em seus parâmetros a cada etapa de treinamento. O maior desafio, nesse caso, é rastrear a região dos *embeddings* das classes quando o gerador de *embedding* é atualizado e o modelo não possui mais acesso às amostras dessas classes.

O uso de classes prototípicas é uma técnica amplamente empregada em diversos métodos dentro da categoria Distance-based. Uma classe prototípica é um *embedding* que representa a tendência central das amostras de uma mesma classe. A Figura 3.7 ilustra um exemplo onde os *embeddings* em amarelo claro e vermelho claro representam gatos e cachorros, respectivamente. O ponto amarelo escuro ($\overline{p}_c$) representa a classe prototípica de gatos e é obtido pela média das características dos *embeddings* dos gatos. Da mesma forma, o ponto vermelho escuro ($\overline{p}_d$) representa a classe prototípica dos cachorros e é criado pela média das características dessa classe.

O uso de uma classe prototípica, em vez de armazenar todas as amostras, auxilia o *Incremental Learning* a reduzir os requisitos de memória e a evitar o treinamento em classes com quantidades desbalanceadas de amostras. O primeiro problema é resolvido porque o modelo precisa armazenar apenas um único ponto de dados. O segundo problema foi identi-

ficado por Belouadah & Popescu [2019], que mostraram que os modelos tendem a favorecer as classes mais recentes. Esse viés é causado pela diferença na quantidade de amostras entre as novas classes e as classes aprendidas anteriormente.

A escolha da representação dos *embeddings* é crucial para determinar a capacidade de um classificador em evitar o esquecimento. Quando as representações das amostras de diferentes tarefas estão alinhadas no espaço de *embedding*, o aprendizado de uma tarefa facilita o aprendizado da outra. Em contrapartida, quando são ortogonais, o aprendizado de uma tarefa não afeta a outra. Em um estudo recente, Javed & White [2019] utilizaram um algoritmo de *meta-learning* para treinar um extrator de características que busca representações que maximizam a paralelização ou ortogonalidade dos *embeddings*, permitindo que o classificador defina uma região para as classes de cada tarefa.

No contexto de *few-shot learning*, outra técnica utilizada para definir o espaço de cada classe no espaço de *embedding* é o alinhamento entre representações visuais e representações semânticas. Cheraghian et al. [2021] propuseram um método no qual o modelo recebe uma imagem como entrada e produz um *embedding* que deve estar próximo ao *embedding* da palavra que representa a classe, em vez de gerar um rótulo como saída. Esse método foi testado utilizando os *embeddings* do *Word2Vec* [Church, 2017] e do *GloVe* [Pennington et al., 2014]. Dessa forma, um modelo treinado para lidar com problemas de classificação em linguagem natural pode ser utilizado sem sofrer com *Catastrophic Forgetting*.

Em situações onde o gerador de *embedding* atualiza seus pesos durante o treinamento, o espaço de representação de uma classe pode sofrer alterações. Para lidar com esse problema, o método *Semantic Drift Compensation* (SDC) [Yu et al., 2020] atualiza todos os protótipos ao final de cada sessão de treinamento para identificar a nova região do espaço de *embedding* onde cada classe será alocada. Para calcular essa mudança, o modelo determina a posição dos dados recebidos no espaço de *embedding* antes e depois do treinamento. A diferença entre essas posições é então utilizada para atualizar os protótipos. No entanto, é importante destacar que o SDC assume que todos os *embeddings* das classes continuam agrupados em torno de seus respectivos protótipos, o que nem sempre é garantido. Dessa forma, embora essa técnica possa ser eficaz em determinados cenários, ela pode não fornecer atualizações precisas para os protótipos em todas as situações.

O aprendizado incremental não supervisionado de representações utilizando a técnica *MixUp* [Liang et al., 2018] apresentou resultados promissores, superando os obtidos com o aprendizado incremental supervisionado. Uma possível explicação para esse fenômeno é que o aprendizado não supervisionado gera uma superfície de perda mais suave [Madaan et al., 2022]. Essa abordagem foi testada com os métodos *Barlow Twins* [Zbontar et al., 2021] e *SimSiam* [Chen & He, 2021]. O classificador principal utilizado nessa categoria baseia-se na similaridade do cosseno. Para aprimorar esse classificador, duas melhorias foram propostas:

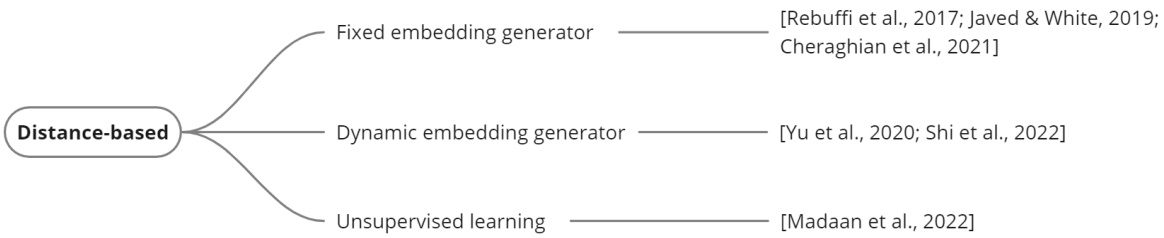


Figura 3.8. Visão geral das principais abordagens na categoria Distance-based. Nesta abordagem, os classificadores baseados em cosseno são os mais utilizados, assumindo que os *embeddings* da mesma classe permanecem agrupados na mesma região. Figura extraída do *survey* de Aleixo et al. [2024].

(i) geração de múltiplas perspectivas de cada amostra e (ii) uso de uma função de perda contrastiva. Na primeira abordagem, cada amostra é utilizada para gerar múltiplas visualizações de si mesma, aplicando rotações. Na segunda abordagem, as amostras geradas são utilizadas como âncoras em uma função de perda contrastiva [Wu et al., 2021].

A função de perda contrastiva encoraja as representações de uma classe ficarem próximas das outras amostras da mesma classe, e manter uma distante das outras classes de no mínimo margem pré-definida. Já a *triplet loss* incentiva a distância positiva máxima (entre um par de representações com o mesmo rótulo) a ser menor que a distância negativa mínima mais a constante de margem pré-definida [Schroff et al., 2015].

Quando um modelo é treinado utilizando a abordagem de *Joint Training*, onde todas as classes são treinadas simultaneamente, observa-se uma melhor definição das fronteiras de decisão dentro do espaço de *embedding*, em comparação ao treinamento incremental. Em geral, essa abordagem tende a produzir uma separação mais robusta entre as classes, aumentando a capacidade do modelo de distingui-las e melhorando sua generalização. Portanto, Shi et al. [2022] propõem uma função objetivo que busca forçar a fronteira de decisão gerada pelo treinamento incremental a imitar aquela obtida pelo *Joint Training*.

A Figura 3.8 apresenta uma visão geral da categoria *Distance-based*. Uma de suas vantagens em relação a outras abordagens é que essa técnica pode expandir o número de classes conhecidas sem modificar a estrutura do modelo, concentrando-se exclusivamente no espaço de *embedding*. No entanto, ainda não está claro quantas classes podem ser efetivamente suportadas por um espaço de *embedding* sem gerar confusão. Na próxima seção, introduziremos uma categoria que busca resolver esse problema identificando submódulos dentro do modelo capazes de mitigar o *Catastrophic Forgetting* (CF).

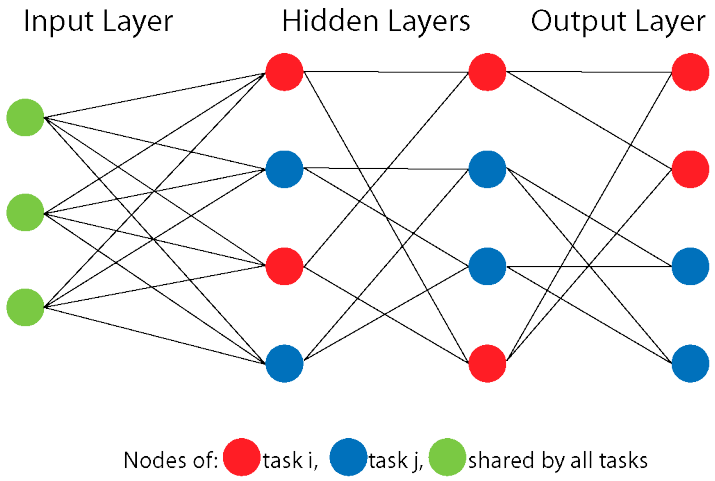


Figura 3.9. Abordagem de sub-rede compartilhada. Exemplo de duas sub-redes dentro da rede principal. Figura extraída do *survey* de Aleixo et al. [2024].

3.1.3 Sub-redes

Sub-redes considera que o problema de esquecimento catastrófico é causado pela sobreposição de conhecimento nas conexões de rede neural. Ao aprender uma nova tarefa, padrões previamente adquiridos podem ser substituídos pelos novos padrões de treinamento, o que pode levar ao esquecimento (*Catastrophic Forgetting* - CF). Por exemplo, um modelo pode utilizar uma conexão neural como um estímulo positivo para melhorar seu desempenho ao aprender a tarefa inicial e, posteriormente, ao aprender uma segunda tarefa, essa mesma conexão pode precisar ser usada como um estímulo negativo, o que pode comprometer o desempenho do modelo na tarefa anterior. Para lidar com esse problema, algumas abordagens procuram codificar o conhecimento necessário para cada tarefa em seções separadas do modelo, minimizando a sobreposição, como ilustrado na Figura 3.9. Ao considerar uma rede neural como um modelo de alta capacidade com conexões redundantes, torna-se viável atualizar seletivamente subconjuntos específicos de neurônios durante o treinamento de cada nova tarefa. Esses subconjuntos, conhecidos como *Sub-networks*, consistem em conjuntos distintos de neurônios dentro do modelo principal, e o objetivo é minimizar a sobreposição entre eles.

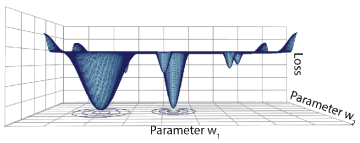
Goodfellow et al. [2014a] conduziram um estudo pioneiro sobre o problema do CF. Eles realizaram uma série de experimentos com diferentes funções de ativação e técnicas de regularização para determinar quais combinações eram menos propensas ao CF. Descobriram que a combinação da função de ativação *ReLU* [Nair & Hinton, 2010] com a regularização por *Dropout* [Hinton et al., 2012] era a mais eficaz. O *Dropout* funciona desativando aleatoriamente um subconjunto de conexões em cada iteração de treinamento, forçando o

modelo a aprender utilizando apenas as conexões ativas restantes. Isso incentiva a codificação redundante de padrões em diferentes subconjuntos de conexões, permitindo que, mesmo que algumas conexões sejam alteradas em tarefas futuras, caminhos alternativos ainda possam ser usados para resolver a tarefa inicial. Esse estudo sugere que um modelo pode ser composto por múltiplos submodelos. Assim, escolher um bom regime de treinamento pode reduzir o CF em uma ampla gama de modelos [Mirzadeh et al., 2020], combinando técnicas como *dropout*, *weight decay*, taxa de aprendizado, tamanho do lote e otimizadores. Vale destacar que essas estratégias podem ser aplicadas em conjunto com outras técnicas discutidas neste estudo.

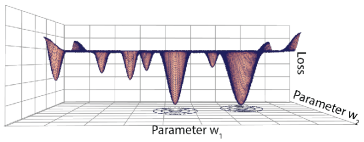
A categoria de *Sub-networks* pode ser dividida em duas abordagens: **Hard Sub-networks** e **Soft Sub-networks**. As abordagens baseadas em *Hard Sub-networks* buscam identificar um subconjunto de parâmetros que representa o conhecimento aprendido de uma tarefa específica e então congelá-los. Consequentemente, quando o modelo precisa aprender uma nova tarefa, ele ajusta um conjunto diferente de neurônios para formar pequenas redes dentro do modelo principal. Isso resulta em um submodelo para cada tarefa aprendida, cada um com desempenhos variáveis. No entanto, como um novo subconjunto de parâmetros precisa ser congelado para cada tarefa treinada, o número de tarefas que o modelo pode aprender é limitado, já que o tamanho da rede permanece constante.

Semelhante às *Hard Sub-networks*, as *Soft Sub-networks* também selecionam um conjunto de parâmetros importantes para a tarefa t_i. No entanto, em vez de congelar esses parâmetros permanentemente, um termo de penalização é adicionado à função de perda para restringir alterações nos parâmetros críticos da tarefa t_i. Isso garante que a rede minimize as mudanças nesses parâmetros enquanto ajusta os valores dos parâmetros da tarefa t_{i+1}. Assim, as *Soft Sub-networks* buscam superar as limitações de capacidade observadas nas abordagens de *Hard Sub-networks*, permitindo pequenas alterações nos neurônios ao longo do treinamento de novas tarefas.

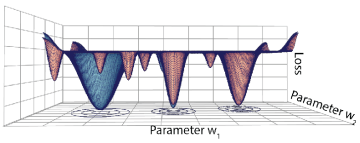
A adição de um termo de penalização à função de perda modifica a paisagem da função de perda. Sabe-se que essa paisagem contém múltiplos mínimos locais que resultam em níveis satisfatórios de acurácia para uma determinada tarefa, como ilustrado na Figura 3.10 [Lee et al., 2017a]. Ao adicionar um termo de penalização, cada submodelo pode ser forçado a atingir um mínimo local que ofereça um desempenho ideal para cada tarefa simultaneamente. Dessa forma, nas *Soft Sub-networks*, diferentes mínimos locais podem corresponder a diferentes tarefas solucionáveis por cada submodelo. O objetivo do treinamento bem-sucedido de uma *Soft Sub-network* é aproximar os mínimos locais de cada submodelo, permitindo que os parâmetros de um submodelo sejam reutilizados por outro. Por exemplo, após aprender a tarefa t_i e iniciar o aprendizado da tarefa t_{i+1}, a rede é forçada a encontrar uma solução aceitável para t_{i+1} que seja o mais próxima possível da solução encontrada para



((a)) *Loss landspace* da tarefa t_i.



((b)) *Loss landspace* da tarefa t_{i+1}.



((c)) Sobreposição das *loss landspaces* das duas tarefas.

Figura 3.10. Paisagem de perda de duas tarefas. Cada tarefa possui mais de um conjunto de pesos possíveis (Parâmetro w_1 e Parâmetro w_2) que apresentam resultados similares em termos de acurácia média do modelo. No lado direito (c), observa-se a sobreposição das duas paisagens de perda, onde o ponto estrela representa uma região de pesos que é uma boa solução para ambas as tarefas. Figura extraída do *survey* de Aleixo et al. [2024].

t_i. Os trabalhos mais relevantes para essas duas abordagens serão discutidos nas próximas subseções.

3.1.3.1 Soft sub-redes

Soft sub-redes permite que duas tarefas compartilhem a mesma conexão, permitindo que a última adapte seus parâmetros de tal forma que o impacto na primeira tarefa seja mínimo. Nessa categoria, cada submodelo se especializa em resolver uma tarefa específica, compartilhando parâmetros (pesos) com outros submodelos, o que ajuda a evitar as limitações de capacidade observadas nas *Hard sub-redes*. No entanto, o subconjunto de parâmetros importantes para a tarefa t_i, que será compartilhado pelos submodelos das tarefas subsequentes $t_{i+1}, t_{i+2}, \ldots$, não pode sofrer grandes alterações durante o treinamento dessas novas tarefas. Para manter esses parâmetros úteis para a tarefa original, eles devem ser identificados antes do início do treinamento da tarefa t_{i+j}, onde j representa um valor inteiro positivo.

A *Fisher Information Matrix* (FIM) foi utilizada por Kirkpatrick et al. [2017] para determinar quais parâmetros são mais importantes para resolver a tarefa t_i. Essas informações são então incorporadas à função de perda como um termo de regularização, impedindo que esses parâmetros sejam significativamente alterados durante o treinamento da tarefa t_{i+1}. Esse método é conhecido como *Elastic Weight Consolidation* (EWC) e é uma das abordagens mais amplamente citadas na área de *Soft Sub-networks*.

O EWC utiliza uma matriz de importância dos pesos da *Rede Neural Profunda* (RNP) para penalizar alterações nos pesos mais relevantes. Para isso, emprega a matriz de informação de Fisher (F), que mede a quantidade de informação que uma variável aleatória observável X carrega sobre um parâmetro desconhecido ω, do qual a probabilidade de X depende. Sendo $\theta = X$ os pesos do modelo, o FIM é usado para regularizar a função de perda durante

o treinamento quando o modelo precisa aprender uma nova tarefa, conforme a equação:

$$\mathscr{L}(\theta) = \mathscr{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2, \tag{3.1}$$

onde $\mathscr{L}_B(\theta)$ representa uma função de perda arbitrária, λ é um hiperparâmetro que define a rigidez da regularização, θ_i são os pesos do modelo que estão sendo ajustados, $\theta^*_{A,i}$ são os pesos ótimos encontrados no incremento i, e $F(\cdot)$ quantifica o impacto causado pela alteração dos pesos com base no FIM.

Hong et al. [2019] introduziram o *Predictive EWC* como uma melhoria do algoritmo EWC. No *Predictive EWC*, antes que o modelo comece a aprender uma nova tarefa t_{i+1}, ele processa os novos dados de treinamento utilizando o modelo atual. Essa etapa de pré-processamento permite que o modelo identifique e descarte amostras que ele já consegue classificar corretamente, mantendo apenas aquelas em que apresenta menor acurácia. Como resultado, um conjunto de dados refinado é formado, assumindo que certas amostras não são mais necessárias para o modelo, pois ele já aprendeu a manipulá-las de maneira eficaz. Esse refinamento reduz o tempo de treinamento, tornando o processo mais eficiente.

Seguindo essa mesma linha de pesquisa, Kobayashi [2018] sugerem que, quando os dados de diferentes tarefas apresentam diferenças substanciais na distribuição, uma abordagem de regularização mais agressiva deve ser aplicada aos parâmetros importantes. Para resolver essa questão, o autor propõe uma função de perda condicional chamada *Check Regularization*, que facilita o agrupamento de parâmetros com base em cada tarefa, permitindo ainda o compartilhamento de parâmetros entre tarefas similares. Ao incorporar a *Check Regularization*, o modelo pode adaptar sua estratégia de regularização para considerar variações nas distribuições das tarefas, melhorando seu desempenho em cenários diversos.

O algoritmo EWC assume que a *Fisher Information Matrix* (FIM) é uma matriz diagonal. No entanto, essa suposição não se sustenta na prática. Para lidar com essa limitação, uma técnica proposta por Liu et al. [2018] pode ser empregada. A ideia central é rotacionar os parâmetros do modelo de forma que: (i) a saída do modelo permaneça inalterada; e (ii) a FIM calculada a partir dos gradientes se torne aproximadamente diagonal. Essa abordagem baseada em rotação demonstrou maior acurácia em comparação ao método original do EWC. No entanto, essa melhoria vem acompanhada de um aumento no tempo de processamento.

Embora essas abordagens tenham mostrado avanços significativos, o uso de técnicas de regularização fortes pode potencialmente prejudicar a capacidade do modelo de adquirir novos conhecimentos. Além disso, existem duas desvantagens notáveis associadas a esses métodos. Primeiramente, eles não consideram a presença de camadas de *batch normalization*, que são amplamente utilizadas em redes neurais profundas (DNNs). Em segundo lugar, essas técnicas envolvem um alto custo computacional durante o pós-processamento, especi-

almente ao calcular a FIM. Essas limitações devem ser levadas em consideração ao aplicar essas técnicas em cenários práticos.

Na pesquisa sobre *Catastrophic Forgetting* (CF), a influência das camadas de *batch normalization* é frequentemente negligenciada, pois, em vez de melhorar a precisão do modelo, elas podem degradá-la quando a distribuição dos dados muda ao longo do tempo. No entanto, para lidar com essa limitação, uma solução promissora chamada *Continual Normalization* (CN) foi proposta por Pham et al. [2022]. A camada CN realiza a normalização espacial do mapa de características utilizando *group normalization*. Ao incorporar camadas CN nos modelos de CF, os efeitos adversos da *batch normalization* em distribuições de dados variáveis podem ser mitigados, preservando assim a precisão do modelo ao longo do tempo.

Para superar o alto custo computacional associado ao cálculo da *Fisher Information Matrix* (FIM) durante o pós-processamento, a técnica de *Synaptic Intelligence* (SI) oferece uma solução viável. O SI envolve a criação de uma matriz Ω_k que representa a importância de cada parâmetro. Essa matriz é calculada durante a sessão de treinamento por meio da aplicação de pequenas perturbações aos valores dos parâmetros e da observação da degradação resultante na solução, conforme descrito por Zenke et al. [2017]. Parâmetros que causam maior impacto negativo são considerados mais importantes.

De forma semelhante, Jung et al. [2020] também empregam uma matriz de importância dentro do arcabouço do SI, porém com duas diferenças principais. Primeiramente, em vez de calcular a matriz durante o treinamento, eles a computam durante a inferência, reduzindo assim a sobrecarga computacional no pós-processamento. Em segundo lugar, eles anulam os pesos de saída dos nós considerados irrelevantes. Quando o modelo precisa aprender uma nova tarefa, esses pesos anulados são re-inicializados aleatoriamente. Essa abordagem permite que o modelo adquira novos conhecimentos ao longo do tempo, ajustando sua arquitetura e suas conexões sinápticas para acomodar os requisitos das novas tarefas.

Outra abordagem para lidar com o custo computacional do pós-processamento envolve considerar todos os parâmetros como importantes e tentar mantê-los dentro da mesma região definida pela primeira tarefa. Isso pode ser alcançado por meio do método de *Monte Carlo Variational Inference* [Blundell et al., 2015]. Ao minimizar a variação de todos os parâmetros do modelo utilizando a divergência de Kullback-Leibler, o modelo busca preservar seus valores ao longo das diferentes tarefas [Nguyen et al., 2018].

Para refinar ainda mais essa abordagem, Ritter et al. [2018] introduziram o conceito de *Online Laplace Approximation with Kronecker*. Por meio dessa técnica, eles estimaram a distribuição *a posteriori* dos parâmetros. Essa estimativa não apenas considera a correlação de um parâmetro com ele mesmo na próxima tarefa, mas também leva em conta sua correlação com outros parâmetros. Ao incorporar essas dependências, o modelo consegue capturar relações mais sutis entre os parâmetros, resultando em uma estimativa mais precisa

da distribuição *a posteriori*.

Inspirado pelo *Learn without Forgetting* (LwF) — discutido na próxima seção —, o *Learn without Memorizing* (LwM) utiliza a *Knowledge Distillation* (KD) [Hinton et al., 2015] para mitigar o esquecimento durante as sessões de treinamento [Dhar et al., 2019]. Enquanto o LwF é focado no cenário de *Task Incremental Learning*, o LwM se concentra no *Class Incremental Learning*. Isso significa que ele não cria um classificador inteiro para cada tarefa, apenas adiciona uma saída para cada nova classe no classificador existente, e regulariza os caminhos internos do modelo para classificar cada nova tarefa. Para alcançar isso, a KD é aplicada nas duas últimas camadas do modelo: a camada de *embedding* e a camada de classificação.

A técnica de *Knowledge Distillation* (KD) trabalha com o conceito de um modelo *professor* e um modelo *aluno*. No aprendizado contínuo, o professor é representado pelo modelo $\mathscr{M}^*$, enquanto o aluno é o modelo $\mathscr{M}$. O professor é uma versão do modelo congelada após o aprendizado da última tarefa, e o aluno é a versão corrente, que sofre alterações em seus pesos devido ao treinamento na tarefa mais recente.

Para regularizar o modelo, durante o treinamento, o professor processa a entrada e gera uma saída o^*, denominada *soft label*. Em seguida, o aluno processa a mesma entrada e gera uma saída o. O objetivo da distilação do conhecimento é fazer com que a saída do aluno se aproxime da saída do professor. Formalmente, o KD pode ser expresso pela seguinte equação:

$$\mathscr{L}(\theta) = \mathscr{L}_B(\theta) + \sum_{k=1}^{|K|} \lambda (o_k - o_k^*)^2, \tag{3.2}$$

onde $\mathscr{L}_B(\theta)$ representa uma função de perda arbitrária, λ é um hiperparâmetro que define a rigidez da regularização, K é o conjunto de amostras, e o_i^* e o_i são as saídas geradas pelos modelos professor e aluno, respectivamente. Essa saída pode ser o rótulo (*softmax*) ou qualquer camada intermediária do modelo, desde que sejam equivalentes.

Ao aplicar regularizações desse tipo no aprendizado incremental, deparamo-nos com o dilema plasticidade-estabilidade [Mermillod et al., 2013]. Esse dilema é caracterizado pela escolha do valor de λ nas regularizações. Quanto maior for a regularização, mais conhecimento o modelo retém das tarefas anteriores, mas, em contrapartida, menor é sua capacidade de aprender novas tarefas. Analogamente, um menor valor de regularização permite que o modelo se adapte mais facilmente a uma nova tarefa, porém aumenta a probabilidade de esquecimento das tarefas anteriores.

A maior parte das pesquisas nessa categoria foca em penalizar ou controlar mudanças nos parâmetros para aumentar a estabilidade do modelo, mas isso pode reduzir sua plasti-

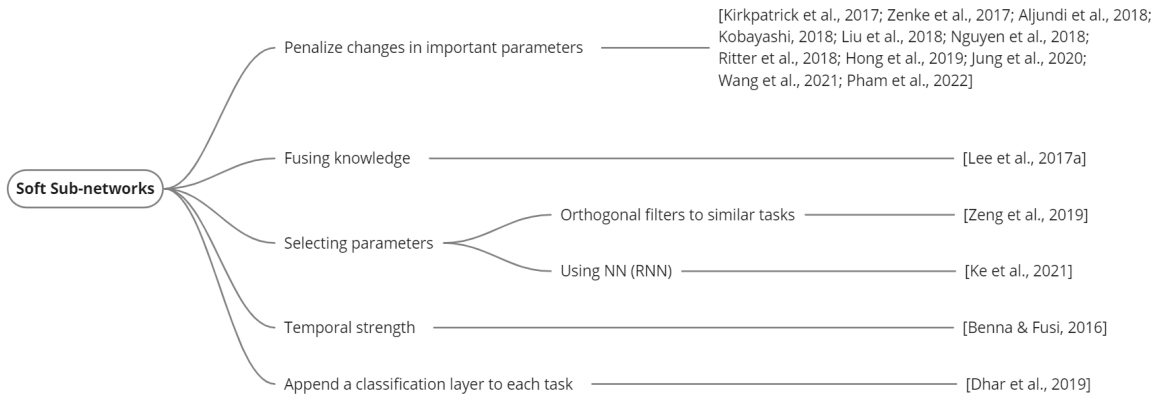


Figura 3.11. Visão geral das principais abordagens na categoria de *Soft Sub-networks*. Nessas sub-redes, as tarefas podem compartilhar parte dos parâmetros selecionados para resolver outras tarefas. Os trabalhos discutidos nesta seção estão agrupados por suas abordagens. Figura extraída do *survey* de Aleixo et al. [2024].

cidade, dificultando a aprendizagem de novas tarefas. Embora preservar o conhecimento prévio seja importante, é essencial equilibrar estabilidade e adaptabilidade para adquirir novos conhecimentos de forma eficiente, ao mesmo tempo que se retém o que já foi aprendido. A Figura 3.11 fornece uma visão geral dessa categoria.

Um desafio com esses métodos é que frequentemente há um desvio na distribuição dos dados de entrada entre as tarefas. Se as tarefas são semelhantes, como nas divisões do MNIST, todos os métodos deste grupo conseguem lidar com o problema, pois o desvio é pequeno e as amostras pertencem ao mesmo domínio. A escolha do método a ser utilizado depende do tempo de treinamento, sendo *Synaptic Intelligence* (SI) ou *Incremental Moment Matching* (IMM) uma melhor escolha para tempos de treinamento mais curtos. No entanto, quando as tarefas vêm de domínios diferentes, o desvio é grande, e nenhum desses métodos consegue lidar com o problema. Isso ocorre porque resolver a tarefa t_{i+1} requer mudar parâmetros críticos que foram aprendidos para resolver a tarefa t_i. Para enfrentar esse problema, outra abordagem de pesquisa, conhecida como *Hard Sub-networks*, foi proposta. Nessas sub-redes, as conexões utilizadas para resolver a tarefa t_i são congeladas, e essa abordagem será discutida na próxima seção.

3.1.3.2 Hard sub-redes

Hard sub-redes congelam os parâmetros definidos como importantes para uma tarefa, deixando apenas os outros parâmetros livres para aprenderem novas tarefas. Nessa categoria, o principal objetivo é encontrar um subconjunto de pesos dentro do modelo principal. No entanto, diferentemente das *Soft Sub-networks*, após a seleção do conjunto de pesos relevantes

para uma tarefa t_i, esses pesos não podem ser modificados durante o treinamento das tarefas t_{i+j}, onde j representa um valor inteiro positivo.

Coop et al. [2013] conduziram um estudo pioneiro nessa categoria ao propor as *Fixed Expansion Layer* (FEL) *Networks*. O FEL expande a rede neural original adicionando uma camada extra após cada camada oculta. Essas camadas adicionais são maiores que suas predecessoras e cada neurônio da camada original é conectado apenas a um subconjunto de neurônios na nova camada oculta. Além disso, os pesos dessas novas camadas são congelados, garantindo que nunca sejam alterados durante o treinamento. Essa expansão das camadas impede a sobreposição do conhecimento entre as tarefas. Como a escolha das conexões entre a camada oculta e a camada adicional é aleatória, os autores utilizam um conjunto de redes FEL em um esquema de *ensemble*. Dessa forma, essa técnica não é categorizada como uma abordagem híbrida ou dinâmica, pois a estrutura do modelo é alterada apenas no início do processo.

A seleção de um subconjunto de neurônios de entrada para mitigar o *Catastrophic Forgetting* (CF) foi introduzida por Goodrich & Arel [2014]. Essa abordagem força diferentes regiões do espaço de entrada a serem aprendidas em diferentes nós da rede, evitando a sobreposição do conhecimento. Gepperth et al. [2015] propuseram o uso de *Self-Organizing Maps* (SOM) para correlacionar a entrada com os neurônios da rede. Em cada sessão de treinamento, o SOM é atualizado caso o modelo falhe na predição; caso contrário, apenas a última camada passa por um ajuste fino. Assim, para cada tarefa (representada pela distribuição estatística descoberta pelo SOM), um novo subconjunto de parâmetros é selecionado. Um trabalho semelhante foi realizado por Lancewicki et al. [2015], mas, em vez de utilizar SOM, os autores empregaram o algoritmo não supervisionado *KNN*. Para separar os neurônios, foi utilizada a distância de Mahalanobis com uma matriz de covariância esparsa gerada por um estimador de *Shrinkage* [Ledoit et al., 2012]. Outros pesquisadores concentraram seus esforços na escolha de parâmetros da rede completa, devido às limitações de recursos na camada de entrada.

Uma abordagem baseada em Algoritmos Genéticos (GA) foi utilizada por Fernando et al. [2017] para encontrar o subconjunto mínimo de parâmetros necessário para aprender cada tarefa sequencialmente. No início do treinamento, cada cromossomo representa um subconjunto de parâmetros. O modelo congela os demais parâmetros e aplica um mecanismo tradicional de aprendizado baseado em gradientes. Quando a solução estabiliza, os parâmetros utilizados são reservados para essa tarefa. Dessa forma, quando uma nova tarefa precisa ser aprendida, o modelo só pode utilizar os parâmetros que ainda não foram reservados. É essencial considerar soluções que utilizem o menor número possível de pesos, pois, caso contrário, a primeira tarefa poderia consumir todos os recursos disponíveis.

A técnica de *pruning* (podagem de pesos) introduzida por Han et al. [2017, 2015]

demonstrou que um modelo pode manter uma acurácia semelhante utilizando um número reduzido de parâmetros, desde que o processo de *pruning* e re-treinamento seja executado incrementalmente. Esse processo consiste em anular certos pesos e re-treinar a rede no mesmo conjunto de dados até que a acurácia atinja um limite pré-definido. Os pesos anulados são então considerados como podados, permitindo que a rede resolva a tarefa apenas com os pesos restantes. Seguindo essa ideia, Guo et al. [2018] utilizaram esse processo para selecionar o subconjunto mínimo de parâmetros necessário para cada tarefa. Além disso, aplicaram a norma L_1 na fase de treinamento para acelerar o processo de *pruning*. Os resultados mostraram que é possível resolver algumas tarefas utilizando menos de 10% dos recursos totais da rede.

Masse et al. [2018] demonstraram que, ao selecionar aleatoriamente apenas 20% dos pesos para cada tarefa (congelando os outros 80%), uma rede neural pode aprender uma sequência de até 100 tarefas sem sofrer significativamente com *Catastrophic Forgetting* em tarefas como *Permuted-MNIST*. Para que essa abordagem seja bem-sucedida, é necessário combiná-la com técnicas de regularização de pesos, como *Elastic Weight Consolidation* (EWC) [Kirkpatrick et al., 2017] ou *Synaptic Intelligence* (SI) [Zenke et al., 2017].

O PackNet utiliza uma máscara binária de pesos para cada tarefa [Mallya & Lazebnik, 2018]. Isso gera uma sobrecarga de armazenamento à medida que o número de tarefas aumenta. Para minimizar o espaço necessário para armazenar as máscaras, os autores propuseram o uso de uma máscara incremental. Ou seja, se o modelo utiliza os parâmetros p_1, p_2 e p_3 para aprender a primeira tarefa, ele congela esses parâmetros e os reutiliza, em conjunto com outros (não congelados), para aprender uma nova tarefa. Para criar essa máscara, os autores também utilizaram o método de *pruning* iterativo de Guo et al. [2018].

Serrà et al. [2018] estenderam o PackNet ao implementar um mecanismo de atenção chamado *Hard Attention to the Task* (HAT). Esse mecanismo acopla uma porta à saída dos neurônios, controlando o fluxo de informação transmitido. A máscara é aprendida junto com a tarefa, utilizando uma função sigmoide como porta de controle. Como a sigmoide produz valores entre zero e um, foi introduzido um mecanismo para forçar os resultados a assumirem apenas valores binários (zero ou um), gerando assim a máscara definitiva. O HAT foi testado em oito tarefas sequenciais e conseguiu aprendê-las sem apresentar *Catastrophic Forgetting* (CF) na tarefa inicial.

Adel et al. [2020] aprimoraram essa técnica ao equipar cada nó do modelo com três variáveis adicionais para controlar a máscara. A primeira variável é binária e define se o nó pode ou não ser adaptado, enquanto as outras duas determinam a magnitude da adaptação. Essas variáveis são aprendidas via inferência variacional.

Em alguns cenários, tarefas podem compartilhar uma máscara, ou pelo menos parte dela, sem causar CF nas tarefas anteriores, permitindo que a nova tarefa seja aprendida. Essa

abordagem é válida para tarefas similares. Ke et al. [2020] definiram que duas tarefas t_i e t_{i+j} (onde j é um inteiro positivo) são consideradas semelhantes se houver uma transferência positiva de conhecimento de t_i para t_{i+j}. Para avaliar isso, um modelo treinado do zero na tarefa t_{i+j} é comparado com um modelo pré-treinado na tarefa t_i e ajustado (*fine-tuned*) para a tarefa t_{i+j}. Se o modelo ajustado apresentar melhor desempenho, considera-se que houve uma transferência positiva de conhecimento. Os autores também utilizaram o HAT para encontrar as máscaras mais apropriadas.

O Piggyback [Mallya et al., 2018] representa uma abordagem diferente das técnicas anteriores, pois não seleciona um subconjunto de parâmetros específicos para cada tarefa t_i. Em vez disso, o Piggyback encontra uma máscara que, ao ser aplicada a um modelo pré-treinado congelado (como um ResNet [He et al., 2016] treinado no conjunto de dados ImageNet [Russakovsky et al., 2015]), resulta em uma solução para a tarefa t_i. Assim, a máscara define quais parâmetros do modelo devem ser utilizados para cada tarefa, mas os pesos do modelo nunca são alterados. No entanto, quando o modelo base (backbone) foi treinado em um contexto muito diferente das tarefas avaliadas, os resultados foram insatisfatórios, pois os novos padrões não puderam ser aprendidos.

Zhai et al. [2020] estenderam esse conceito para *Generative Adversarial Networks* (GANs), criando um banco de filtros que pode ser reutilizado em tarefas futuras em vez de criar novos filtros do zero. Mais recentemente, Xue et al. [2022] demonstraram que essas máscaras também podem ser utilizadas em modelos de *Vision Transformers* (ViT) para evitar o CF.

As redes modulares são compostas por múltiplos módulos interconectados densamente dentro de cada grupo, mas com conexões esparsas entre os módulos [Clune et al., 2013, Lipson, 2007, Wagner et al., 2007]. No contexto de Reinforcement Learning (RL), redes modulares foram empregadas para mitigar o CF [Ellefsen et al., 2015]. No entanto, há uma carência de comparações diretas com *benchmarks* estabelecidos, pois os autores propuseram um novo problema de RL. Eles projetaram um conjunto distinto de neurônios de entrada para cada tarefa, conforme ilustrado na Figura 3.12. Os neurônios vermelhos recebem amostras exclusivamente de uma tarefa, enquanto os neurônios azuis recebem amostras de outra tarefa. Essa organização permite que a estrutura interna do modelo ative seletivamente diferentes subconjuntos de neurônios para cada tarefa.

Uma variação das redes modulares, conhecida como *Diffusion-based Neuromodulation*, foi introduzida por Velez & Clune [2017] para identificar módulos específicos de cada tarefa dentro do modelo. Os autores estabeleceram pontos de conexão dentro da arquitetura para regular a modularidade. Em sua avaliação, dois pontos de entrada adicionais foram introduzidos, um para cada tarefa. O primeiro ponto recebia um estímulo positivo, enquanto o outro ponto era suprimido ao processar amostras da primeira tarefa. No caso da

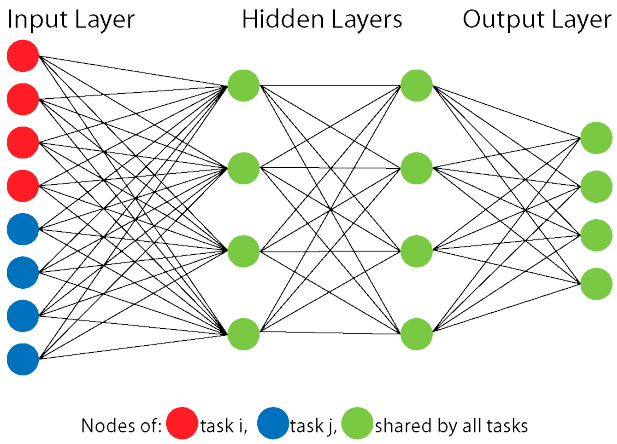


Figura 3.12. Figura adaptada de Ellefsen et al. [2015]. No topo, observa-se a camada de entrada. Cada tarefa possui quatro valores como entrada, portanto, os quatro primeiros neurônios foram utilizados no treinamento da tarefa t_i, enquanto os quatro subsequentes foram usados na tarefa t_{i+1}. Figura extraída do *survey* de Aleixo et al. [2024].

segunda tarefa, os estímulos eram invertidos. No entanto, os autores testaram seu método apenas no problema específico que propuseram, e sua extensão para outras tarefas apresenta desafios, pois exige um design e configuração manuais do modelo.

Uma solução potencial para lidar com a limitação de capacidade de um modelo é fornecer-lhe mais recursos do que o necessário e permitir que cada camada contribua para a tomada de decisão em todas as tarefas, criando assim um modelo baseado em *ensemble*. Um exemplo dessa abordagem é o *Incremental Adaptive Deep Model* (IADM), que aproveita as saídas dessas camadas como entrada para uma rede rasa. Essa rede emprega um mecanismo de atenção para realizar a predição final [Yang et al., 2019].

A Figura 3.13 fornece uma visão geral da categoria de Hard sub-redes. A maioria dos estudos nesta categoria foca em explorar o subconjunto ideal de parâmetros para resolver múltiplas tarefas de maneira eficiente. É fundamental notar que esse subconjunto deve ser o menor possível, devido a limitações de recursos computacionais. Conforme o tempo avança, todos os recursos disponíveis (parâmetros do modelo) acabam sendo utilizados, tornando o modelo incapaz de aprender novas tarefas sem sofrer o esquecimento catastrófico (CF).

Para mitigar essa limitação, alguns pesquisadores propuseram uma nova categoria de estratégias, conhecidas como Redes dinâmicas, que permitem a expansão do modelo conforme necessário. A categoria de redes dinâmicas será discutida a seguir.

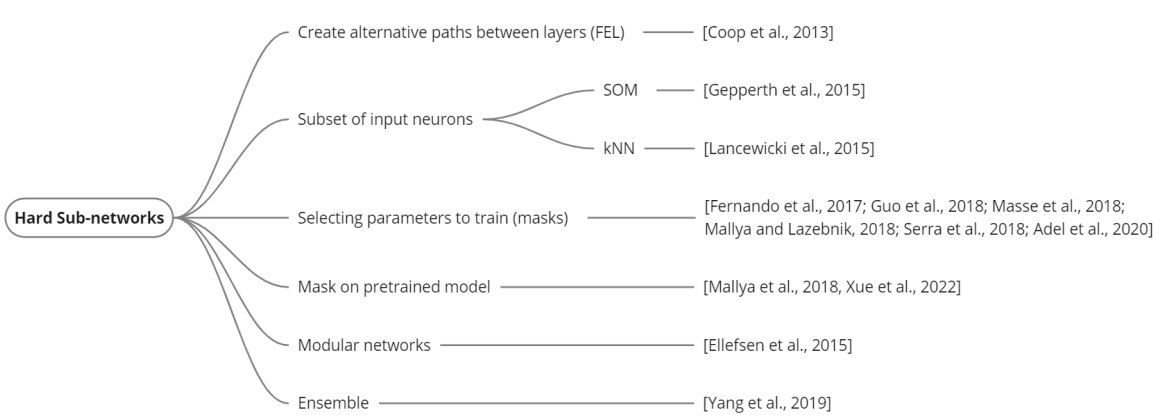


Figura 3.13. Visão geral das principais abordagens na categoria de *Hard Sub-networks*. Nessa abordagem, cada tarefa tem permissão para modificar apenas um subconjunto único de parâmetros. Os trabalhos discutidos nesta seção estão agrupados de acordo com suas abordagens. Figura extraída do *survey* de Aleixo et al. [2024].

3.1.4 Redes Dinâmicas

Redes dinâmicas parte do princípio de que o problema do esquecimento catastrófico (CF) é causado pela sobreposição de conhecimento dentro das conexões da rede neural. Além disso, sugere que um número fixo de recursos (parâmetros) não permitirá que um modelo aprenda novas tarefas indefinidamente sem esquecer as anteriores. Dessa forma, as técnicas de Redes Dinâmicas contornam esse problema ao permitir a expansão da capacidade do modelo sempre que necessário.

O trabalho pioneiro nessa linha foi um sistema analítico baseado na pseudoinversa da matriz de entrada, proposto por Serre [2002]. Esse sistema foi utilizado para identificar os parâmetros ideais ao adicionar um novo neurônio na camada de saída. Embora essa abordagem possa mitigar o problema de CF, ela é limitada a redes que seguem a arquitetura *Extreme Learning Machine* (ELM) [Huang et al., 2006]. As ELMs são redes neurais compostas por uma camada de entrada, uma camada oculta e uma camada de saída. Como essa abordagem depende da estrutura específica das ELMs, não pode ser aplicada diretamente a arquiteturas mais complexas de redes neurais profundas (DNNs).

No contexto das DNNs, Rusu et al. [2016] iniciaram a investigação nessa categoria ao propor as Progressive Neural Networks (PNN). A principal ideia do PNN é expandir progressivamente a rede neural à medida que novas tarefas são aprendidas, criando uma nova rede para cada tarefa sem modificar os parâmetros das redes anteriores.

No PNN, toda a estrutura da rede é replicada para cada nova tarefa, inicializando os pesos da nova rede com os mais recentemente treinados. Além disso, cada camada l_k da tarefa t_i está conectada à entrada da camada l_{k+1} na tarefa t_{i+1}. Dessa forma, a rede recém-criada

pode reutilizar conhecimento das redes anteriores, facilitando a aprendizagem incremental e reduzindo a necessidade de recomeçar o treinamento do zero.

A Figura 3.19 ilustra a arquitetura desse *framework*. Para cada nova tarefa, uma nova rede é adicionada ao conjunto, e conexões laterais são criadas entre as camadas correspondentes das diferentes redes. Essas conexões laterais permitem que a nova rede aproveite representações aprendidas anteriormente, enquanto evita modificações nas redes congeladas. Essa estratégia impede a degradação do desempenho nas tarefas já aprendidas, mitigando o esquecimento catastófico.

Apesar de ser eficaz para mitigar o esquecimento catastófico, o PNN apresenta uma limitação significativa em termos de custo computacional e consumo de memória. Como uma nova rede completa é adicionada para cada nova tarefa, o modelo cresce linearmente em relação ao número de tarefas aprendidas. Esse crescimento pode rapidamente se tornar inviável em aplicações práticas, especialmente quando um grande número de tarefas precisa ser aprendido. Esse problema motivou pesquisas posteriores para minimizar o aumento de recursos, explorando estratégias de expansão modular e crescimento sublinear [Ostapenko et al., 2021].

Alternativas mais eficientes, como Expert Gate [Aljundi et al., 2017] e DualMemory [Lee et al., 2017b], foram desenvolvidas para reduzir a redundância na alocação de novos recursos e melhorar a escalabilidade do modelo. Além disso, abordagens mais recentes aplicadas em Transformers [Douillard et al., 2022, Hu et al., 2023] investigam o uso de módulos especializados e o compartilhamento seletivo de parâmetros para controlar o crescimento da rede ao longo do tempo.

Para mitigar a crescente taxa de consumo de recursos, Ramesh & Chaudhari [2022] adotam uma abordagem contrastante e propõem a utilização de uma única DNN para extração de características, acompanhada por múltiplas redes rasas, cada uma dedicada a uma tarefa específica. Ao empregar um extrator de características compartilhado entre todas as tarefas, os autores sugerem a adoção de um modelo pré-treinado ou o congelamento do extrator de características após o treinamento da primeira tarefa. Durante uma sessão de treinamento, apenas a rede rasa especializada na tarefa atual é atualizada. A inferência é realizada agregando as saídas de todos os classificadores por meio de uma técnica de média. Os autores defendem que essa abordagem é eficaz por atuar como um classificador Boosting, garantindo um desempenho aprimorado na classificação enquanto utiliza os recursos de forma mais eficiente.

Li & Hoiem [2018] propõem adicionar apenas uma nova camada *softmax* ao modelo para cada nova tarefa. Esse método é conhecido como Learn without Forgetting (LwF) e exige um algoritmo de treinamento específico. Antes de iniciar uma nova sessão de treinamento, o modelo cria um conjunto de *soft labels* para cada dado. Um *soft label* é a distribui-

ção de probabilidade sobre múltiplas classes possíveis e, nesse caso, é gerado pelas camadas *softmax* do modelo. Esse rótulo auxilia o modelo na generalização e é utilizado para aplicar *Knowledge Distillation* (KD) [Hinton et al., 2015] durante a sessão de treinamento. Quando as camadas geram os rótulos, são conhecidas como Professores. Por outro lado, quando utilizam esses rótulos para adquirir conhecimento, são denominadas *Students*. Durante a sessão de treinamento, a técnica de KD é aplicada nessas camadas *softmax*, enquanto os dados da tarefa atual são utilizados para treinar a nova camada *softmax* adicionada.

O método descrito anteriormente utiliza um extrator de características compartilhado para gerar uma representação de *embedding* antes da camada *softmax*. No entanto, uma limitação significativa dessa abordagem é que o *embedding* pode ser alterado à medida que a rede aprende novas tarefas. Como consequência, torna-se difícil manter o conhecimento original, uma vez que a representação pode sofrer modificações substanciais durante o treinamento. Isso representa um desafio para o desenvolvimento de algoritmos eficazes que evitem o *Catastrophic Forgetting* (CF) seguindo essa metodologia [Rannen et al., 2017].

Outros trabalhos focaram na pesquisa de métricas para indicar onde e/ou quando o modelo precisa ser expandido. Cai et al. [2018] propuseram uma métrica que mede o quão bem uma nova tarefa é ajustada pelos filtros em cada camada de uma CNN já treinada. Essa métrica é chamada de Averaged Response Variance (ARV) e é calculada como:

$$\text{ARV} = \frac{1}{N} \sum_{n=1}^{N} R_n, \tag{3.3}$$

onde N é o número de filtros em cada camada e R é a variância das saídas dos filtros após o processamento do conjunto de dados da nova tarefa. Os autores propuseram modificar a estrutura do modelo com base nessa métrica. Quando o valor de ARV é alto, o número de filtros é aumentado para ajustar o conhecimento da nova tarefa.

Ashfahani & Pratama [2019] demonstram que outras métricas podem ser utilizadas para essa decisão. Eles propuseram o uso de métricas de *overfitting* e *underfitting* para modificar o número de neurônios em uma camada e o número de camadas, se necessário. Embora os resultados de acurácia sejam comparáveis entre diferentes abordagens, o crescimento do modelo pode variar dependendo da tarefa e da ordem em que é aprendida. Assim, é essencial avaliar cada cenário individualmente para determinar qual abordagem apresenta a menor taxa de crescimento.

Em vez de depender de uma métrica fixa, alguns pesquisadores optaram por um método treinável, no qual uma entidade auxiliar toma as decisões. O Neural Architecture Search (NAS) foi utilizado para expandir um modelo até encontrar uma estrutura ótima para suportar uma nova tarefa [Li et al., 2019]. Três operações foram consideradas para essa busca:

i) adicionar novos filtros à camada, congelando os anteriores; ii) reutilizar os filtros previamente congelados; e iii) criar uma nova camada do zero. Em paralelo, Xu & Zhu [2018] treinaram um agente de aprendizado por reforço de forma *end-to-end* para decidir quando e onde adicionar nós em uma rede neural para acomodar novas tarefas. Embora esses métodos geralmente exijam mais tempo para encontrar a versão final do modelo, eles conseguem otimizar melhor sua estrutura, reduzindo seu tamanho final.

Métodos hierárquicos também podem ser utilizados para determinar quando expandir um modelo. O Tree-CNN [Roy et al., 2020] emprega uma estrutura em árvore para organizar sua arquitetura. Cada nó pode fornecer uma predição ou encaminhar a entrada para outro nó tomar a decisão. Cada nó possui sua própria CNN para realizar classificações e, sempre que uma nova tarefa é apresentada e o nó não consegue diferenciá-la das já aprendidas, um nó filho é criado. O principal problema dessa abordagem é que, se um erro ocorre em um nível intermediário, a inferência final também será comprometida.

Outra linha de pesquisa dentro dessa categoria envolve o reuso de um modelo congelado, replicando apenas uma parte específica da arquitetura original. Diferente de outras abordagens, essa técnica não exige modificações na estrutura geral da DNN. Em vez disso, apenas os pesos das partes relevantes do modelo são atualizados, enquanto os demais permanecem inalterados. Esse método permite preservar de forma eficiente o conhecimento previamente adquirido, ao mesmo tempo em que possibilita a assimilação de novas informações em tarefas específicas. Imai & Nobuhara [2018] propuseram uma nova DNN com uma estrutura idêntica à de um modelo pré-treinado, mas com pesos inicializados aleatoriamente. Durante o treinamento, o modelo pode decidir se utilizará a camada l_k da DNN pré-treinada ou da recém-criada. Assim, apenas uma parte da rede precisa ser armazenada, tornando essa técnica aplicável a qualquer arquitetura. Os resultados demonstraram que as primeiras camadas costumam ser aproveitadas da DNN pré-treinada, pois contêm características fundamentais dos dados. No entanto, os autores não testaram o método em uma sequência de tarefas com mudança de domínio.

Com base nos achados de Imai & Nobuhara [2018], que evidenciaram que as camadas iniciais de uma CNN aprendem características de baixo nível, Zacarias & Alexandre [2018b] propuseram o SeNA-CNN. Esse modelo especial de CNN contém cinco camadas convolucionais. Quando precisa aprender uma nova tarefa, ele simplesmente replica as camadas a partir da terceira, compartilhando as duas primeiras entre todas as tarefas. Diferente do PNN, não há conexões entre camadas de diferentes tarefas. Tanto PNN quanto SeNA-CNN não possuem um mecanismo nativo para identificar a qual tarefa uma amostra pertence. Para resolver essa questão no SeNA-CNN, Zacarias & Alexandre [2018a] propuseram uma melhoria onde uma CNN auxiliar, equipada com uma camada *softmax*, determina qual ramo da rede utilizar durante a inferência.

Uma abordagem ainda mais eficiente em termos de uso de recursos foi proposta por Xiong et al. [2018]. Nesse método, apenas estatísticas (média e desvio padrão) das últimas camadas são armazenadas, e durante a inferência, os pesos são gerados dinamicamente com base na tarefa. Para determinar a qual tarefa a amostra pertence, um *autoencoder* é treinado para cada tarefa. Embora essa técnica economize recursos da rede neural, ela adiciona complexidade ao sistema, exigindo a criação de novas estruturas e armazenamento de informações adicionais. Vale ressaltar que, quando diferentes tarefas compartilham as primeiras camadas do modelo, há um viés em favor da primeira tarefa aprendida. Esse problema é menos pronunciado quando as tarefas pertencem a domínios similares, mas pode levar a resultados insatisfatórios quando aplicadas a domínios distintos.

Diversos estudos têm explorado o uso de métodos e estratégias dentro da categoria de Redes Dinâmicas para lidar com o *Catastrophic Forgetting* (CF) em cenários mais desafiadores e variados. Por exemplo, Lee et al. [2020] trabalham com o Unbounded Task Incremental Learning, propondo um *framework* que cria uma nova rede, chamada de Expert, para lidar com classes desconhecidas. À medida que os dados chegam em um fluxo contínuo, quando uma amostra não é reconhecida por nenhum Expert existente, ela é armazenada temporariamente em uma memória efêmera. Quando essa memória atinge sua capacidade, um novo Expert é criado e treinado com esses dados, que são então descartados após o treinamento.

No contexto de Redes Geradoras Adversárias (GANs), Verma et al. [2021] propõem integrar cada parâmetro da rede em duas partes: i) uma parte compartilhada; e ii) uma parte específica da tarefa. A primeira é fixada após a primeira tarefa ou obtida de um modelo pré-treinado, enquanto a segunda é criada para cada nova tarefa. Dessa forma, a soma das duas partes permite que o modelo continue a produzir bons resultados. Para garantir que as partes específicas de cada tarefa sejam distintas, os autores aplicam restrições baseadas na divergência de Kullback-Leibler.

Um dos desafios típicos enfrentados pelas técnicas de Redes Dinâmicas é a necessidade de identificar a qual tarefa uma determinada amostra pertence. Esse problema também ocorre em alguns modelos do grupo de Sub-redes. Algumas abordagens propõem modelos equipados com um seletor de tarefas embutido, mas nem todos os métodos adotam essa solução. Por isso, alguns autores focam no desenvolvimento de métodos que possam funcionar como oráculos para decidir a qual tarefa cada amostra pertence. Gepperth & Gondal [2018] sugerem o uso de um K-Nearest Neighbors (KNN) como oráculo, criando apenas uma nova camada de saída para cada nova tarefa. Embora essa abordagem seja eficiente em termos de recursos, os testes apresentados foram limitados a conjuntos de dados Permutation MNIST, onde as estatísticas de entrada das tarefas são altamente similares, ou seja, não há uma mudança significativa na distribuição entre as tarefas.

Em resumo, os modelos dessa categoria possuem a capacidade de aumentar seu tama-

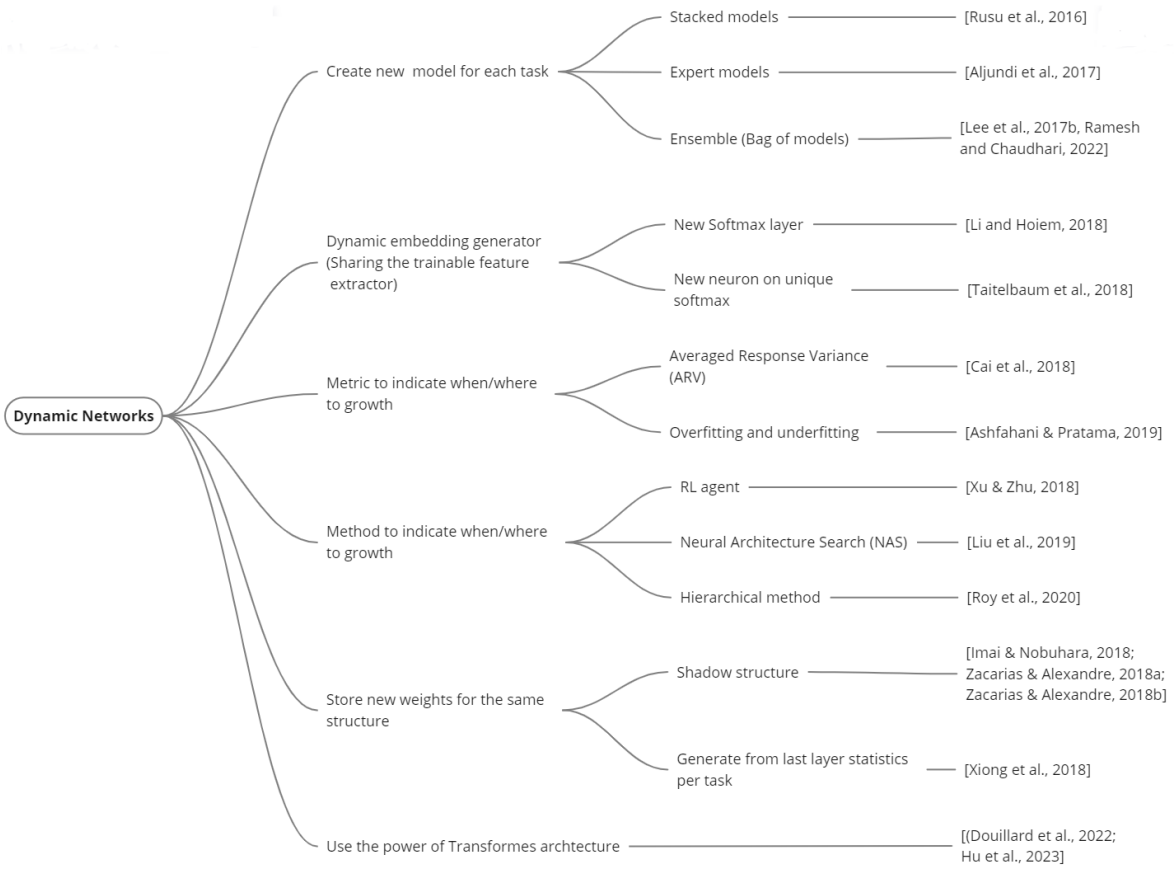


Figura 3.14. Visão geral das principais abordagens na categoria de Redes Dinâmicas. Em Redes Dinâmicas, é possível expandir os recursos (parâmetros) para acomodar novos conhecimentos. Os trabalhos discutidos nesta seção estão agrupados de acordo com suas abordagens. Figura extraída do *survey* de Aleixo et al. [2024].

nho para acomodar novos conhecimentos, com algumas abordagens modificando sua estrutura e outras criando estruturas auxiliares. A Figura 3.14 fornece uma visão geral dessa categoria. O principal objetivo desses modelos é aprender novas tarefas enquanto minimizam o crescimento da arquitetura. No entanto, alguns autores aceitam um crescimento polinomial do modelo para otimizar o aprendizado de novas tarefas. Assim como na categoria de Sub-redes, a identificação da tarefa associada a cada amostra continua sendo um desafio nas abordagens de Redes Dinâmicas. Essa área de pesquisa ainda está em evolução e apresenta grande potencial para aprimorar os métodos existentes. No entanto, uma desvantagem notável dessa categoria é a necessidade de espaço para armazenar o modelo, tornando-a menos adequada para cenários como computação em dispositivos de borda (*edge computing*) ou aplicações móveis, onde as restrições de espaço são críticas.

A seguir, discutiremos métodos que combinam estratégias de múltiplas categorias, aproveitando as vantagens de cada abordagem.

3.1.5 Híbridos

Os **métodos híbridos** combinam duas ou mais técnicas descritas anteriormente para lidar com o esquecimento catastrófico (CF) de maneira mais eficaz. Dessa forma, aproveitam os pontos fortes de diferentes estratégias para obter um melhor desempenho. Algumas combinações são relativamente simples de implementar, como a incorporação de métodos de rehearsal com outras técnicas, e geralmente resultam em melhorias na acurácia das tarefas previamente aprendidas.

Além disso, a flexibilidade das abordagens híbridas permite que essas soluções sejam adaptadas a requisitos específicos, alcançando um equilíbrio entre a mitigação do esquecimento e a aquisição de novos conhecimentos.

Aprimorando Métodos Baseados em Distância com Rehearsal: Abordagens baseadas em distância enfrentam um desafio significativo ao tentar acompanhar os protótipos das classes quando o extrator de características sofre modificações nos pesos durante o treinamento. Essas mudanças nos pesos fazem com que os vetores de protótipos se desloquem para diferentes regiões no espaço de características, o que pode gerar confusão no processo de tomada de decisão do classificador final. Para mitigar esse problema, técnicas de rehearsal podem ser utilizadas armazenando um subconjunto de amostras e seus respectivos vetores de características. Com isso, os vetores de características das amostras armazenadas podem ser monitorados durante a aprendizagem de novas tarefas, permitindo a detecção de deslocamentos de distância causados pelas alterações nos pesos. O ICaRL foi pioneiro na combinação de métodos de rehearsal e abordagens baseadas em distância para mitigar o *Catastrophic Forgetting* (CF), garantindo representações precisas das classes mesmo diante de modificações nos pesos e deslocamentos dos protótipos [Rebuffi et al., 2017].

Como demonstrado por Liu et al. [2020a], o rehearsal pode ser combinado de maneira eficaz com métodos baseados em distância e *Knowledge Distillation* (KD) em um contexto de *meta-learning*, permitindo a adaptação a novas tarefas enquanto minimiza a interferência com as tarefas já aprendidas. Essa integração é complementada pelo uso de um *coreset*, que ajuda a manter *embeddings* consistentes das amostras armazenadas. O modelo proposto por Liu et al. [2020a] é composto por dois principais componentes: um extrator de características e um classificador. No início da sessão de treinamento, uma cópia temporária do extrator é criada com seus parâmetros congelados. Durante o treinamento, tanto o extrator congelado quanto o ativo produzem *embeddings* idênticos para os mesmos dados de entrada. Esse alinhamento é essencial para garantir que o classificador alcance alta precisão na predição de novas classes durante o processo de *fine-tuning*. Todo o processo ocorre dentro do *loop* interno do *meta-learning*.

Além disso, em um cenário de Aprendizado Incremental Online, onde o modelo não

possui conhecimento do identificador da tarefa e deve aprender em uma única passagem pelos dados, uma abordagem baseada em distância combinada com uma técnica de *Mini-rehearsal* pode ser utilizada para mitigar o CF. Para lidar com a limitação de poucas amostras por classe, a técnica MixUp, introduzida por [Liang et al., 2018], pode ser empregada para data augmentation. Ao combinar essas estratégias, incluindo métodos baseados em distância, *KD*, *Mini-rehearsal* e *MixUp*, o modelo pode se adaptar a novas tarefas, reter conhecimento prévio e mitigar o CF enquanto supera as limitações desse cenário.

Métodos baseados em distância se beneficiam do uso de classificadores de distância coseno, que demonstraram maior resistência ao CF em comparação com classificadores *softmax* [Davari et al., 2022, Simon et al., 2021]. No entanto, essa vantagem se mantém apenas quando os *embeddings* das mesmas classes podem ser efetivamente agrupados no espaço latente e permanecerem suficientemente distantes dos clusters de outras classes. Para atender a esse requisito, Mai et al. [2021] propõem armazenar um *coreset* de amostras e aplicar *Contrastive Loss* durante o treinamento. Ao utilizar o *coreset* e a *Contrastive Loss*, busca-se incentivar a organização automática dos *embeddings*, facilitando uma melhor separação entre classes. Assim, o modelo pode estruturar melhor os *embeddings* no espaço latente, resultando em maior separação entre classes e maior resistência ao CF.

Em configurações altamente desafiadoras, como o Aprendizado Incremental Online combinado com um cenário de Tarefas Não Limitadas, o uso de técnicas de rehearsal torna-se essencial para preservar o conhecimento anterior em métodos baseados em distância. Para lidar com esse desafio, De Lange & Tuytelaars [2021] propõem uma abordagem baseada em memória composta por dois componentes: amostras e protótipos. O armazenamento em memória gerencia eficazmente o fluxo de dados desbalanceado característico desses cenários. A suposição subjacente é que os protótipos mudarão ao longo do tempo conforme novas informações forem recebidas. Assim, uma nova função de perda é introduzida para guiar o extrator de características na geração de *embeddings* que se aproximem dos protótipos da classe correspondente, ao mesmo tempo em que se afastam de outros protótipos. Esse mecanismo melhora a separação entre classes e aumenta a capacidade do modelo de reter conhecimento prévio. Conforme novas amostras substituem as anteriores na memória, os protótipos evoluem dinamicamente, adaptando-se à distribuição dos dados em constante mudança. Essa atualização dinâmica dos protótipos garante que o modelo permaneça atualizado e seja capaz de lidar com tarefas evolutivas e cenários de aprendizado ilimitado de maneira eficaz.

A análise das relações geométricas entre classes pode ser explorada sob diferentes perspectivas por meio do uso de Espaços de Curvatura Mista, conforme proposto por Gao et al. [2023]. Essa abordagem consiste em criar espaços distintos para cada tarefa e projetar amostras em todos os espaços relevantes durante a inferência. Ao utilizar um *coreset*, o

objetivo é garantir que novas projeções não introduzam confusão na classificação em espaços previamente aprendidos. De maneira semelhante, Ma et al. [2023] empregam uma técnica de projeção para *embeddings* de amostras, porém utilizando um Diagrama de Voronoi como espaço de referência. Especificamente, quando uma nova classe surge, uma célula dentro do diagrama é particionada. Uma parte da célula é dedicada à retenção do conhecimento já adquirido, enquanto a outra acomoda a nova classe. Esse processo, facilitado pelo uso de um *coreset*, envolve decisões estratégicas sobre a seleção das células a serem divididas, garantindo a preservação de uma estrutura organizada que mitiga os desafios impostos pelo CF.

Em cenários onde o número total de classes é conhecido previamente, mas as classes precisam ser aprendidas de forma sequencial, é possível predefinir uma região distinta para cada classe e treinar apenas uma camada de projeção, conforme proposto por Yang et al. [2023]. É importante destacar que o modelo gerador de *embeddings* permanece fixo durante todo o processo, podendo ser um modelo pré-treinado ou congelado após a primeira tarefa. Como a camada de projeção necessita de atualizações a cada nova tarefa, a inclusão de um *coreset* torna-se essencial para preservar o conhecimento sobre como projetar cada classe corretamente em seu respectivo espaço, enfrentando de maneira eficaz o desafio do CF.

Aprimorando Métodos Baseados em Distância com Redes Dinâmicas: Abordagens híbridas que combinam Redes Dinâmicas com métodos Baseados em Distância oferecem outra solução eficaz para mitigar o *Catastrophic Forgetting* (CF). Um dos melhores exemplos dessa abordagem é o Encoder-Based Lifelong Learning, que cria uma nova camada de classificação para cada nova tarefa e acompanha as mudanças nos *embeddings* das classes [Rannen et al., 2017]. Nesse método, um *autoencoder* é treinado para cada tarefa, permitindo a preservação dos *embeddings* e possibilitando a fixação da camada de classificação das tarefas antigas por meio da *Knowledge Distillation* (KD).

O uso de *autoencoders* nessa abordagem tem dois propósitos principais. Primeiro, permite que os *embeddings* de uma classe permaneçam estáveis ao longo da vida útil do modelo. Segundo, os *autoencoders* podem atuar como oráculos, auxiliando o classificador no processo de aprendizado contínuo do modelo. A seleção da tarefa é baseada nas distâncias entre os *embeddings* preservados, garantindo uma classificação precisa e específica para cada tarefa.

Ao combinar Redes Dinâmicas com métodos Baseados em Distância, e explorando a abordagem Encoder-Based Lifelong Learning, o modelo consegue se adaptar a novas tarefas enquanto retém o conhecimento adquirido anteriormente. Essa abordagem híbrida apresenta uma solução robusta para cenários de aprendizado contínuo, mitigando efetivamente o CF. No entanto, como essa técnica depende do treinamento de um *autoencoder* separado para cada tarefa, o consumo de memória do modelo aumenta à medida que novas tarefas são

introduzidas. Dessa forma, para que essa abordagem seja viável, a estrutura do *autoencoder* precisa ser significativamente menor do que a do modelo original.

Aprimorando Redes Dinâmicas com Rehearsal: Uma das principais limitações dos métodos Redes Dinâmicas está no seu elevado crescimento em termos de recursos computacionais. Para mitigar e controlar essa expansão, uma abordagem eficaz é a manutenção de um *coreset*, que permite ao modelo modificar certos parâmetros sem comprometer o conhecimento adquirido sobre classes anteriores. Dessa forma, o modelo pode continuar aprendendo novas tarefas com uma demanda reduzida de recursos, como um número menor de neurônios e camadas.

Lüders et al. [2017] propõem o uso de uma Evolvable Neural Turing Machine (ENTM) [Greve et al., 2016]. Nesse modelo, uma fita de memória é adicionada para armazenar informações contextuais. Essa fita possui duas cabeças: uma para leitura e outra para escrita, ambas controladas por uma rede neural. Para definir essa rede neural, foi utilizada a abordagem Neuroevolution of Augmenting Topologies (NEAT) [Stanley & Miikkulainen, 2002], que emprega algoritmos evolucionários para aumentar a complexidade da rede neural ao longo do tempo. Dessa forma, o modelo inicia com uma arquitetura rasa e evolui progressivamente até encontrar uma solução satisfatória. Além disso, a ENTM armazena informações contextuais que permitem à rede neural lembrar de conceitos aprendidos anteriormente. A Figura 3.15 ilustra a arquitetura proposta. Esse modelo é híbrido, combinando a ENTM como um sistema complementar de aprendizado e o módulo NEAT, que permite o crescimento dinâmico da rede neural.

Entretanto, esse método foi testado apenas em um contexto de Reinforcement Learning (RL) e requer uma configuração manual para entrada e saída de cada tarefa. Como entrada, o modelo recebe três informações: i) o novo estado do ambiente; ii) a recompensa recebida (saída do ambiente); e iii) a informação contextual armazenada na fita da ENTM. Já a saída do modelo inclui: i) a ação a ser tomada pelo agente (entrada do ambiente); ii) a nova informação contextual a ser escrita na fita; e iii) os comandos de controle para as cabeças da fita. Além disso, os autores não comparam seus resultados com outros métodos da literatura, apenas demonstram que o modelo pode resolver o problema proposto por Ellefsen et al. [2015].

Castro et al. [2018] expandem o modelo adicionando um novo classificador para cada tarefa e utilizando uma abordagem baseada na seleção por *herding* para manter amostras no *coreset*. Diferente do ICaRL, eles treinam um classificador paramétrico junto ao extrator de características. A Figura 3.16 apresenta esse processo de treinamento. Para cada nova tarefa, uma nova camada de classificação é adicionada ao modelo e treinada com função de perda por entropia cruzada, enquanto as camadas anteriores são treinadas com perda de distilação usando *Knowledge Distillation* (KD). O modelo armazena parte dos dados das tarefas ante-

riores, mas também aumenta sua estrutura ao adicionar uma nova camada de classificação para cada nova tarefa. Para lidar com o problema do desequilíbrio do conjunto de dados, os autores aplicam um ajuste fino (*fine-tuning*) pós-treinamento utilizando amostras da memória e um subconjunto das amostras da tarefa atual, garantindo que todas as classes tenham a mesma quantidade de exemplos.

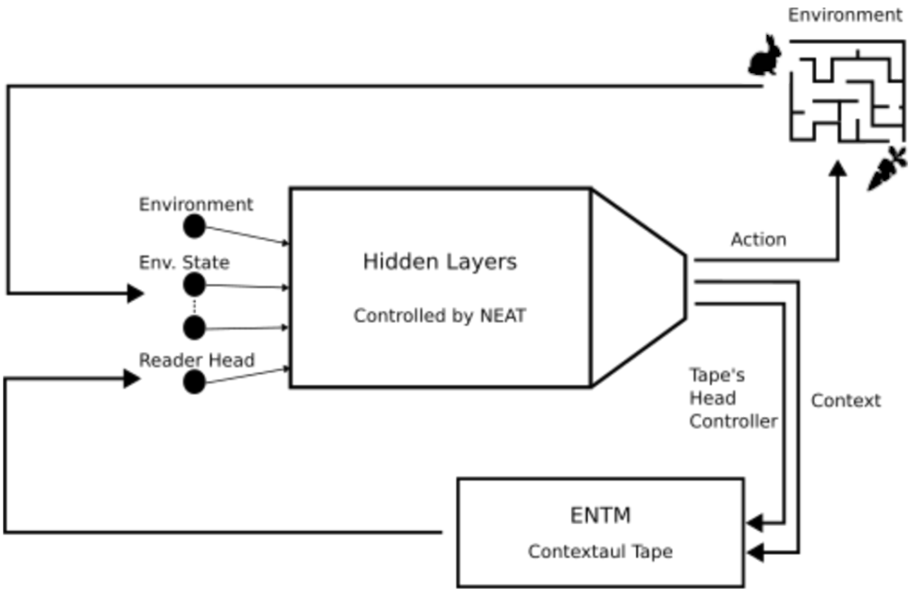


Figura 3.15. Arquitetura proposta por Lüders et al. [2017]. No modelo ENTM, uma rede neural com camadas de entrada e saída fixas controla as ações do agente. As entradas incluem a última recompensa, o estado atual do ambiente e a informação contextual armazenada na fita da ENTM. Além disso, a rede neural gera um novo contexto para ser armazenado na fita da ENTM e controla as cabeças de leitura e escrita. A quantidade e o formato das camadas ocultas são ajustados dinamicamente pelo algoritmo NEAT. Figura extraída do *survey* de Aleixo et al. [2024].

Ostapenko et al. [2019] expandem o Deep Generative Replay (DGR) [Shin et al., 2017] ao adicionar a capacidade de expansão do modelo gerador, criando assim um modelo híbrido que combina Pseudo-rehearsal e Redes Dinâmicas. Após cada sessão de treinamento, os parâmetros do gerador são congelados e, na próxima sessão, novos recursos (camadas e neurônios) são adicionados. Como consequência, é necessário armazenar uma máscara para identificar quais parâmetros pertencem a cada tarefa. Em vez de gerar pseudo-amostras, Joseph & Balasubramanian [2020] propõem o uso de um *Variational Autoencoder* (VAE) para gerar pesos e criar um conjunto de modelos (ensemble), todos com a mesma estrutura. Esses pesos são considerados uma forma de *rehearsal* do conhecimento prévio. Embora o método proposto por Hu et al. [2019] também utilize um módulo gerador de pesos, a técnica de Joseph & Balasubramanian [2020] se diferencia por criar um ensemble de modelos, o que potencializa a retenção do conhecimento passado.

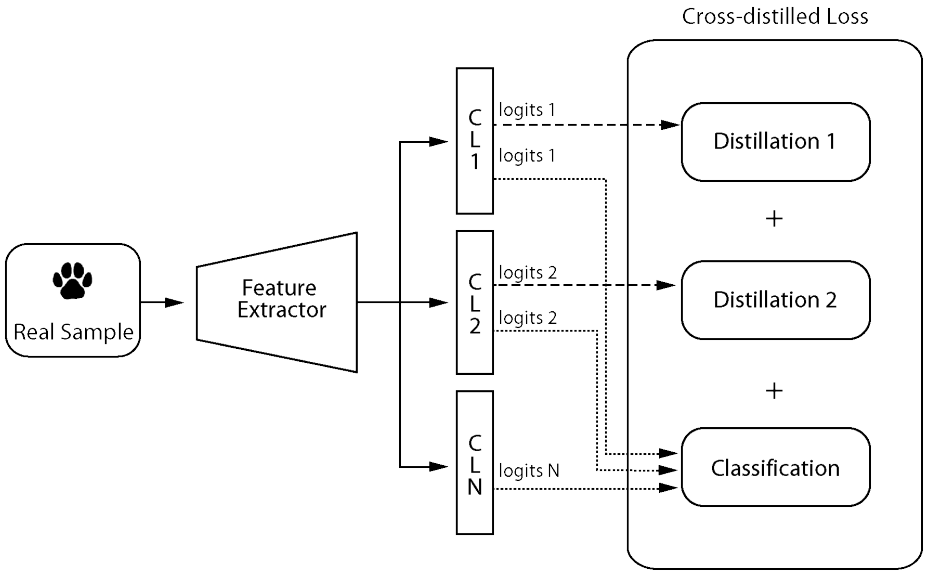


Figura 3.16. Processo de treinamento do modelo de Castro et al. [2018]. As camadas de classificação das tarefas anteriores produzem logits usados para distilação e classificação, enquanto a camada da tarefa atual gera logits exclusivamente para classificação. Figura extraída do *survey* de Aleixo et al. [2024].

Além disso, um agente de Reinforcement Learning (RL) foi utilizado como um método de busca estrutural para expandir o modelo no cenário de Aprendizado Incremental por Tarefas [Qin et al., 2021]. Nesse caso, o ambiente de RL é composto pelos dados da tarefa corrente, um modelo de aprendizado contínuo, um repositório de conhecimento e um *coreset*. O repositório de conhecimento contém os valores dos parâmetros das tarefas anteriores. Assim, o agente usa essas informações para criar um novo modelo aprendiz, minimizando a taxa de esquecimento (validada com o *coreset*) e maximizando a acurácia da tarefa atual.

Outro desafio recorrente nos métodos de Redes Dinâmicas é a necessidade de identificar a qual tarefa uma determinada amostra pertence durante a inferência. O uso de um *coreset* pode auxiliar na mitigação desse problema. A abordagem Dynamically Expandable Representation (DER) Yan et al. [2021] expande o modelo criando um novo extrator de características para cada nova tarefa, mantendo um único classificador. Uma vez aprendido, cada extrator é congelado. O classificador, por outro lado, continua sofrendo modificações em seus parâmetros para aprender novas tarefas, sendo necessário armazenar um *coreset* para evitar o esquecimento catastrófico (CF). Durante a inferência, o classificador recebe como entrada a concatenação das características geradas por todos os extratores aprendidos, fazendo com que a representação de cada amostra cresça à medida que novas tarefas são incorporadas.

Uma alternativa para resolver o problema de identificação de tarefas nos métodos de Redes Dinâmicas é o uso de múltiplas cabeças de classificação, cada uma dedicada a uma

tarefa específica. Durante a inferência, a cabeça com maior confiança na predição pode ser escolhida. Kim et al. [2022] propõem o uso de um *coreset* contendo dados de tarefas anteriores como uma classe negativa dentro do classificador da tarefa atual. Assim, cada classificador pode atuar como um detector de amostras fora da distribuição (*out-of-distribution detector*), facilitando a seleção correta da cabeça correspondente durante a inferência.

Tanto as Redes Dinâmicas quanto as técnicas de Rehearsal apresentam a desvantagem do aumento do consumo de armazenamento em comparação ao modelo original. As Redes Dinâmicas exigem o armazenamento de parâmetros adicionais, enquanto o Rehearsal exige a retenção de amostras no *coreset*. Portanto, o aumento do espaço necessário deve ser cuidadosamente considerado em termos de utilização de recursos e restrições de memória. Zhou et al. [2023] introduzem um *framework* de referência para essa categoria híbrida, levando em conta o consumo total de memória, a adição de novos parâmetros e a presença de amostras no *coreset*. Esse *framework* busca equilibrar cuidadosamente a quantidade de amostras retidas no *coreset* e a expansão do modelo. Por meio de uma análise empírica, os autores observaram que a expansão do modelo nas camadas mais profundas traz benefícios significativos tanto em acurácia quanto em uso de espaço. Notavelmente, esse *framework* supera, em precisão e eficiência de espaço, diversas técnicas discutidas nesta seção.

Aprimorando Sub-redes com Rehearsal: A aplicação do Aprendizado Incremental por meio de técnicas baseadas em Sub-redes é particularmente relevante em cenários com recursos computacionais limitados, como na computação em borda (*edge computing*). A integração de um *coreset*, que armazena um subconjunto de amostras das tarefas anteriores, desempenha um papel crucial na descoberta de subestruturas compartilháveis entre diferentes tarefas [Wang et al., 2022d].

Estudos recentes indicam que pode não ser necessário regularizar todos os pesos do modelo para mitigar o esquecimento catastrófico (CF); em alguns casos, a regularização somente na última camada pode ser suficiente. Essa abordagem, conhecida como regularização funcional, tem sido empregada com sucesso por Titsias et al. [2020] e Pan et al. [2020] utilizando Processos Gaussianos (GPs).

Uma desvantagem da regularização funcional baseada em GPs é a necessidade de armazenar um grande número de amostras das tarefas mais recentes. Como alternativa, uma abordagem híbrida, que combina regularização (identificação de sub-redes na última camada) com *mini-rehearsal*, pode ser uma solução mais eficiente. Vale destacar que a escolha da técnica de *mini-rehearsal* varia entre os estudos, e os resultados obtidos se mostram similares aos de métodos que aplicam regularização em todos os pesos do modelo, mas com uma eficiência computacional significativamente maior.

Outra técnica relevante é a Neural Calibration for Online Continual Learning (NCCL), proposta por Yin et al. [2021]. A NCCL estende o conceito de Elastic Weight Consolidation

(EWC) ao introduzir dois mecanismos para modular os sinais de entrada e saída das Redes Neurais Profundas (DNNs), um processo denominado calibração neuronal. Os parâmetros desse módulo são aprendidos de forma a equilibrar o dilema estabilidade-plasticidade, visando identificar submódulos no modelo que possam ser eficazes para resolver cada tarefa. No entanto, esse método ainda requer o armazenamento de uma parte dos dados de treinamento de cada classe aprendida para garantir um aprendizado eficiente.

Sarfraz et al. [2023] empregam duas técnicas principais para incentivar a formação de Sub-redes em conjunto com Rehearsal: Sparse Activations e Semantic Dropout. - Sparse Activations utilizam a função de ativação k-Winner-Take-All (k-WTA), que permite a propagação seletiva das k unidades com maior ativação para as camadas subsequentes. - Semantic Dropout, por sua vez, induz um dropout seletivo em diferentes neurônios ao aprender novas tarefas, aproveitando os padrões de ativação neuronal. Essa abordagem é particularmente vantajosa ao lidar com tarefas não correlacionadas.

A incorporação de Sub-redes e Rehearsal abre caminhos promissores para o Aprendizado Contínuo em ambientes com recursos computacionais limitados. Essas técnicas permitem a identificação e reutilização de subestruturas específicas para cada tarefa, melhorando a retenção do conhecimento adquirido e facilitando a adaptação a novas tarefas. No entanto, um dos desafios críticos permanece: o armazenamento de dados de treinamento, essencial para garantir a efetividade dessas abordagens.

Expansão de Sub-redes com Redes Dinâmicas: A combinação de Redes Dinâmicas e Sub-redes oferece uma solução para o desafio de recursos limitados enfrentado pelos métodos de Sub-redes. Um dos principais métodos dentro dessa abordagem é a Dynamically Expandable Network (DEN), proposta por Yoon et al. [2018]. O objetivo do DEN é determinar o conjunto mínimo de conexões necessárias para que uma Rede Neural aprenda uma nova tarefa. Caso as conexões existentes sejam insuficientes, o modelo expande dinamicamente seus recursos.

O DEN opera em três fases distintas: treinamento, expansão e mitigação da deriva semântica.

Na fase de treinamento, o modelo passa por um treinamento inicial utilizando regularização L_1, que incentiva a esparsidade dos pesos. Quando uma nova tarefa é introduzida, um nó de saída correspondente a essa tarefa é adicionado. Em seguida, ocorre um pré-treinamento, no qual apenas a última camada permanece não congelada, ajustando unicamente os pesos do nó recém-adicionado. Como as demais partes do modelo também mantêm conexões esparsas, um algoritmo de Busca em Largura (BFS) é empregado para estabelecer um caminho S que conecta cada nó de saída às entradas do modelo. Após essa etapa, todos os neurônios que não pertencem ao caminho S são congelados, e o modelo passa por um treinamento convencional, minimizando a perda associada à tarefa específica.

Na fase de expansão, o modelo verifica se a perda da nova tarefa atingiu um nível aceitável previamente definido. Caso contrário, k neurônios são adicionados a todas as camadas, e uma nova rodada de treinamento é realizada utilizando regularização L_1 para promover a esparsidade e descartar neurônios desnecessários.

Por fim, na fase de mitigação da deriva semântica, o modelo analisa se algum neurônio no caminho original S sofreu mudanças significativas. Se isso ocorrer, indica que o neurônio pode estar sujeito à deriva semântica em relação a alguma tarefa anterior. Assim, esse neurônio é duplicado: uma cópia mantém o valor antigo e é removida do caminho S, enquanto a outra permanece para a tarefa atual.

Esse mecanismo permite que o modelo cresça de maneira controlada, maximizando a reutilização dos neurônios já existentes e evitando o comprometimento das tarefas previamente aprendidas.

Ao ajustar dinamicamente a arquitetura do modelo e expandir seletivamente as conexões, o DEN aborda as limitações de recursos enfrentadas pelos métodos baseados em Sub-redes. Essa abordagem adaptativa permite que o modelo acomode novas tarefas enquanto reduz a interferência no conhecimento previamente adquirido e assegura uma utilização eficiente dos recursos.

Outra abordagem híbrida, proposta por Hung et al. [2019], também lida com o desafio dos recursos limitados ao identificar subestruturas para cada tarefa e expandi-las quando necessário. O método consiste em um processo iterativo composto por três fases. Na primeira fase, o modelo aprende a tarefa utilizando neurônios não congelados, enquanto aplica uma máscara nos neurônios já congelados. A segunda fase envolve a poda iterativa dos neurônios não congelados para otimizar a capacidade do modelo. Esse processo de poda contribui para a alocação eficiente dos recursos, garantindo que apenas as conexões mais relevantes sejam mantidas. Por fim, caso o modelo não atinja a acurácia mínima predefinida, ele passa por um processo de expansão, e o treinamento é reiniciado para a tarefa específica. Essa expansão permite que o modelo adquira recursos adicionais e se adapte às exigências da nova tarefa. Quando o limite de acurácia predefinido é alcançado, a máscara utilizada durante a fase de aprendizado é salva, e os pesos correspondentes são congelados.

Ao aprender, podar e expandir iterativamente as subestruturas do modelo, esse método proporciona uma solução híbrida que gerencia efetivamente as limitações de recursos. Essa abordagem permite que o modelo aloque dinamicamente recursos conforme as necessidades específicas de cada tarefa, preservando o conhecimento adquirido anteriormente.

A combinação das categorias de Sub-redes e Redes Dinâmicas oferece vantagens significativas. De um lado, a categoria de Sub-redes encapsula o máximo de tarefas dentro da mesma estrutura de rede, minimizando o crescimento excessivo do modelo. Por outro lado, a abordagem só permite a expansão quando realmente necessário, mitigando o problema en-

frentado pelas Redes Dinâmicas, onde o tamanho do modelo pode crescer linearmente com o número de tarefas. Portanto, essa abordagem é considerada uma excelente escolha para diversos cenários. No entanto, há ambientes e configurações em que essa técnica não é aplicável, como em Unbound Task Incremental Learning ou em computação de borda, onde os recursos são extremamente limitados. De maneira geral, a acurácia das técnicas dessa categoria é inferior às abordagens baseadas em Rehearsal, que armazenam amostras do passado para reuso no treinamento contínuo.

Combinando Sub-redes, Redes Dinâmicas e Rehearsal: O modelo Else-Net, proposto por Li et al. [2021], combina características das abordagens de Sub-redes, Redes Dinâmicas e Rehearsal. Nesse modelo, cada camada é composta por múltiplos blocos de conhecimento que processam a entrada. Um mecanismo de seleção determina o bloco mais relevante, gerando uma *embedding* para a próxima camada. Quando uma nova tarefa precisa ser aprendida, um bloco temporário é criado. Se o mecanismo de seleção o considerar importante, ele é mantido para futuras tarefas; caso contrário, ele é descartado, e os demais módulos têm seus parâmetros ajustados. Além disso, uma pequena parte (10%) das amostras de cada classe já aprendida é armazenada no *coreset* para mitigar o esquecimento.

Uma diferença notável entre esse modelo e os métodos discutidos anteriormente é que a presença do *coreset* permite a adaptação dos pesos em parte do modelo sem causar um esquecimento significativo. Essa flexibilidade torna o Else-Net aplicável a cenários além do Task Incremental Learning, nos quais o identificador da tarefa é conhecido tanto durante o treinamento quanto na fase de avaliação.

A Figura 3.17 apresenta uma visão geral das abordagens que combinam múltiplas categorias para lidar com o problema do *Catastrophic Forgetting* (CF). Cada categoria possui suas próprias vantagens e limitações. Fica evidente que nenhum modelo pode aprender um número infinito de tarefas com um número fixo de parâmetros. Por outro lado, criar um modelo separado para cada nova tarefa é inviável. Da mesma forma, armazenar todas as amostras previamente treinadas também não é factível devido às limitações de espaço de armazenamento e tempo de re-treinamento. No entanto, preservar um subconjunto de amostras críticas parece ser uma solução eficaz para melhorar a retenção do conhecimento. A comparação entre essas abordagens é desafiadora devido à falta de consenso em relação às métricas, conjuntos de dados e metodologias de teste utilizadas na literatura.

3.2 Cenários de aprendizado

Como foi brevemente mencionado na introdução, existem seis cenários (ou configurações) que os modelos almejar resolver: i) *Domain Incremental Learning*; ii) *Task Incremental*

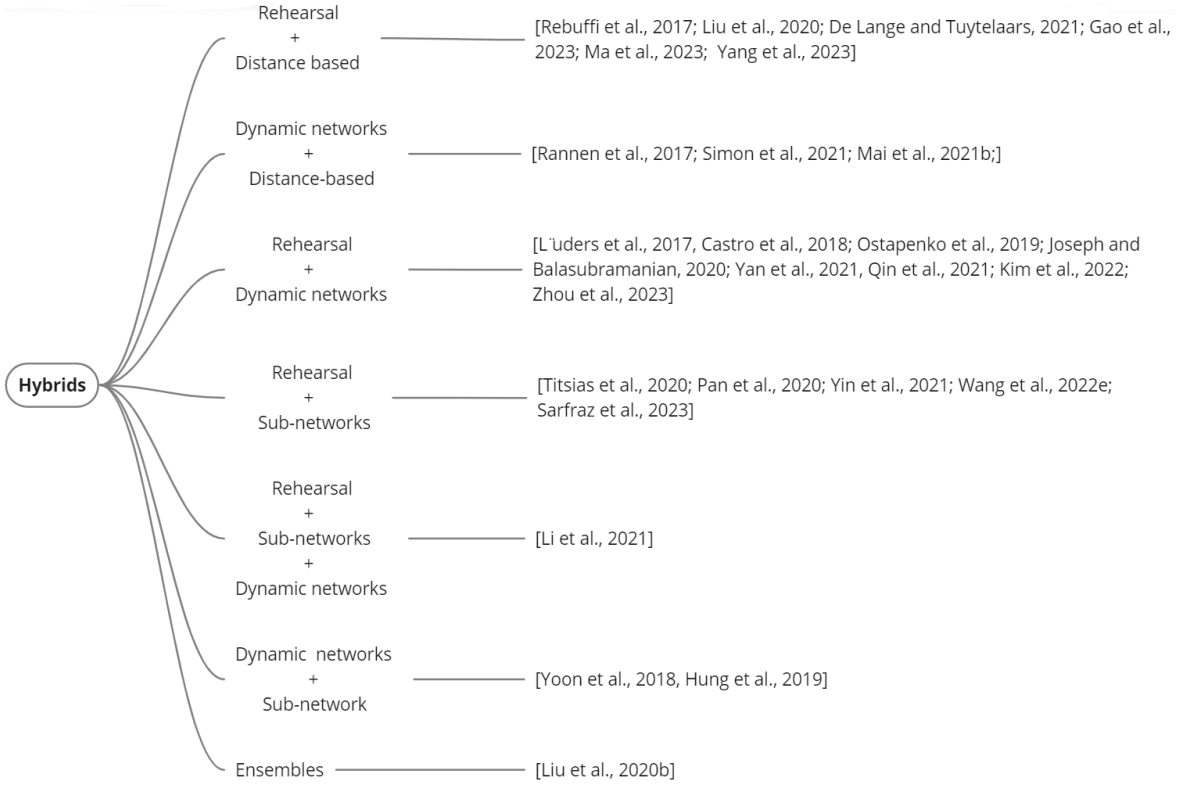


Figura 3.17. Visão geral dos principais métodos na categoria de abordagens híbridas. Os trabalhos discutidos nesta seção estão agrupados por similaridade. Figura extraída do *survey* de Aleixo et al. [2024].

Learning; iii) *Class Incremental Learning*; iv) *Online Incremental Learning*; v) *Unbounded Task Incremental Learning*; e, vi) *Data-free Incremental Learning*.

Domain Incremental Learning é uma configuração onde todas as tarefas devem ter a mesma quantidade de saídas possíveis. Por exemplo, considere um classificador binário, as próximas tarefas também devem ser problemas binários.

Esta configuração representa o aprendizado contínuo de um problema que sofre alterações em suas características ao longo da vida. Por exemplo, em um problema de condução de carro autônomo, um agente pode ser inicializado com um treinamento padrão para considerar a rua de uma cidade, cheia de sinais de trânsito, cidadãos e durante o dia. Em uma próxima tarefa, o agente pode ser obrigado a aprender a dirigir no mesmo cenário, porém, à noite. Em seguida, para aprender a dirigir com neve. Depois, aprende a dirigir em uma estrada, e assim por diante. Neste exemplo, podemos notar que a tarefa é a mesma, dirigir um carro. O que muda é o domínio onde o agente está executando a tarefa.

Essa configuração pode ser aplicada a qualquer outro contexto, como processamento de linguagem natural [Ke et al., 2021]. Também existe uma variação desse *setup*, a adaptação de domínio não supervisionado (UDA) [Rostami, 2021]. Nessa abordagem, apenas

a primeira tarefa contém rótulos. As tarefas subsequentes precisam ser alinhadas com a primeira no aprendizado não supervisionado.

Task Incremental Learning, também conhecida como configuração de várias cabeças, adiciona um novo classificador no topo do modelo para cada tarefa incremental. Todas as tarefas compartilham o corpo do modelo, ou seja, os recursos comuns de todas as tarefas são codificados nas primeiras camadas, enquanto os recursos particulares permanecem em sua parte individual.

Por exemplo, considere o conjunto de dados MNIST dividido em 5 grupos com duas classes cada grupo. Em cada etapa incremental, o modelo deve aprender um grupo usando uma ramificação independente. Como as redes neurais costumam prever qualquer amostra para algumas de suas classes com alta confiança (apesar da amostra não pertencer a nenhuma delas), um grande desafio para essa configuração é decidir qual ramo considerar no momento da inferência. Para resolver este problema os pesquisadores adotam duas soluções: i) sempre informar ao modelo a identificação da tarefa a qual a amostra pertence, portanto, considere apenas aquele ramo; ii) usar um modelo externo como oráculo para decidir de qual tarefa pertence.

Métodos de sub-redes que codificam cada tarefa em algumas conexões do modelo utilizam esta configuração, pois a máscara a ser aplicada para resolver a tarefa correta precisava ser conhecida a priori. As redes dinâmicas também possuem a mesma característica. No momento da inferência, o modelo não pode usar o novo recurso adicionado após a tarefa da amostra, caso contrário o resultado pode não ser o esperado.

Class Incremental Learning difere do *Task Incremental Learning* pelo número de classificadores. Enquanto a primeira possui um classificador para cada tarefa, o segundo possui apenas um classificador. E, a cada passo incremental, o classificador adiciona neurônios em sua camada de saída para representar as novas classes da tarefa de entrada. Por exemplo, considere o conjunto de dados MNIST dividido em 5 grupos com duas classes cada grupo. O classificador do modelo começa com dois neurônios de saída. À medida que uma nova tarefa chega, o modelo expande sua camada de saída com mais dois neurônios.

Um problema inerente aos modelos que consideram essa configuração é o viés para classes de tarefas recentes. Esse problema foi notado por muitos pesquisadores [Belouadah & Popescu, 2020, Hou et al., 2019, Wu et al., 2019]. Isso é causado pela grande quantidade de amostras de novas classes enquanto as antigas são escassas.

Online Incremental Learning é derivado do *Class Incremental Learning*, porém tem mais restrições. Nessa configuração o modelo aprende as classes de cada tarefa em um fluxo contínuo. Isso significa que o modelo não pode executar um treinamento *offline* a cada nova tarefa [Cai et al., 2021]. Para a primeira tarefa o modelo conta com uma grande quantidade de dados e pode proceder quantas iterações de treinamento que forem necessárias. No entanto,

em tarefas subsequentes, o modelo precisa aprender classes com apenas algumas iterações usando uma quantidade escassa de dados.

Devido à baixa quantidade de amostras nessa configuração, Cheraghian et al. [2021] argumenta que é necessário fazer um aumento de dados nos dados de treinamento. Apesar das técnicas tradicionais de *few-shot*, *zero-shot* e *meta-learning* possuírem a capacidade de aprender novas tarefas com poucas amostras e processarem esses dados apenas por algumas iterações, elas não consideram a capacidade de reter o conhecimento de tarefas antigas. Portanto, eles não resolvem o problema do esquecimento catastrófico no aprendizado contínuo. No entanto, eles são considerados a base para os modelos que tentam lidar com o problema nesta configuração [Chen & Lee, 2021, Liu et al., 2020a].

Esta configuração é mais realista para modelos usados em dispositivos de borda que requerem aprendizado contínuo, pois não possuem muito poder computacional. Com o crescimento do campo da Internet das Coisas (IoT), essa configuração está ganhando mais atenção.

Unbounded Task Incremental Learning é direcionado à modelos de classificação e é a configuração mais próxima do que os humanos usam para aprender. A cada passo incremental, o modelo recebe um conjunto de dados para aprimorar seu conhecimento, porém, uma parte desses dados pode pertencer a classes que o modelo desconhecia até então [Caccia et al., 2020, Zheng et al., 2021]. Também é conhecido na literatura como *Task-free Learning* [Aljundi et al., 2019], *Task-agnostic Learning* [Wu et al., 2018b] ou ainda como *Data Incremental Learning* [De Lange & Tuytelaars, 2021].

Os modelos que funcionam nessa configuração devem considerar em cada etapa incremental para identificar amostras de classes desconhecidas, aprender quantas classes foram adicionadas e o padrão para identifica-las no futuro [Buzzega et al., 2020]. Além disso, as amostras de classes conhecidas devem ser utilizadas para reforçar e não esquecer o seu conhecimento. É sempre necessário lidar com o problema do conjunto de dados não balanceado, pois em cada etapa o número de amostras por nova classe é inferido pelo modelo. Não temos muitos trabalhos que visem lidar com essa configuração, no entanto, é uma direção promissora para um modelo alcançar uma capacidade de aprendizagem infinita.

A maioria dos modelos que consideram esta configuração utilizam técnica de classificação não paramétrica baseada na representação das amostras, como distância cosseno. Técnicas usadas para produzir a representação em grupos da mesma classe também são usadas, como a função de perda *triplet loss* [Cha et al., 2021].

Data-free Incremental Learning considera a privacidade a restrição mais relevante para um modelo quando requer aprendizado contínuo [Smith et al., 2021]. Os métodos que usam técnicas de *rehearsal* não podem ser usados nesta configuração. Os pesquisadores que consideram essa configuração defendem que as amostras para aprender uma classe c_m na

tarefa t_i não pode ser usada para manter o conhecimento ao aprender outras classes na tarefa t_{i+1}, a menos que o dono da a amostra autorize.

Por exemplo, considere um sistema de reconhecimento facial. A modelo inicia seu treinamento com fotos de rostos de usuários e aprende a identificar essas pessoas. Quando o treinamento termina, essas imagens não podem ser armazenadas no sistema devido à restrição de privacidade. Portanto, após algum tempo, quando o modelo for obrigado a aprender a identificar novos usuários, o modelo terá acesso apenas às fotos de rosto dos usuários de novas pessoas, e não pode esquecer como identificar os usuários antigos.

Alguns autores [Hu et al., 2021, Michieli & Zanuttigh, 2021, Shmelkov et al., 2017] defendem que mesmo modelos generativos não podem ser usados nesta configuração. Essa restrição ocorre porque quando um modelo generativo é treinado, ele aprende a reproduzir amostras como as originais, portanto violaria a privacidade, por exemplo, um gerador de rostos poderia reproduzir a imagem de um usuário que não permite o uso de sua imagem por o modelo. Essa configuração ganhou muita atenção nos últimos anos devido a requisitos legais sobre privacidade de dados.

3.3 Métricas e Protocolo de Avaliação

Kemker et al. 2018 definiu, com o objetivo de facilitar comparações entre trabalhos da área, um conjunto de três métricas para medir a capacidade de um modelo reter conhecimento enquanto aprende novas tarefas. Essas métricas são apresentadas nas Equações 3.4, 3.5 e 3.6:

$$\Omega_{base} = \frac{1}{T-1} \sum_{i=2}^{T} \frac{\alpha_{base,i}}{\alpha_{ideal}} \tag{3.4}$$

$$\Omega_{new} = \frac{1}{T-1} \sum_{i=2}^{T} \alpha_{new,i} \tag{3.5}$$

$$\Omega_{all} = \frac{1}{T-1} \sum_{i=2}^{T} \frac{\alpha_{all,i}}{\alpha_{ideal}} \tag{3.6}$$

onde T é o número total de tarefas, $\alpha_{new,i}$ é a precisão do teste para a tarefa t_i imediatamente após ser aprendido, $\alpha_{base,i}$ é a precisão do teste na primeira tarefa após i novas tarefas terem sido aprendidas, $\alpha_{all,i}$ é a precisão do teste de todos os dados de teste para as classes vistas até este ponto, e α_{ideal} é a precisão do modelo *offline* no conjunto de base, que é considerado o desempenho ideal. Ω_{base} mede a retenção de um modelo da primeira tarefa, Ω_{new} mede a capacidade do modelo de recuperar imediatamente novas tarefas e Ω_{all} mostra quão bem um

modelo retém conhecimento prévio e adquire novas informações.

Para alguns trabalhos, também é importante avaliar a propagação do conhecimento adiante (*Forward Knowledge Transfer*) [Ke et al., 2020], ou seja, verificar se o conhecimento aprendido na tarefa atual melhora a capacidade de resolver as tarefas futuras. Por exemplo, após o modelo ter aprendido a reconhecer números de 0 a 4, o modelo é avaliado no conjunto de dados de números de 5 a 9, sem treinamento neste segundo conjunto de dados. No entanto, esta avaliação não é considerada em todos as configurações, vide *Unbounded Task Incremental Learning*.

Além de evitar a degradação da acurácia em tarefas antigas, alguns modelos conseguem aperfeiçoar os resultados de uma tarefa t_i após aprender uma tarefa t_{i+j} com j maior ou igual a um. Esse fenômeno é conhecido como propagação de conhecimento inverso (*Backward Knowledge Transfer*) [Ke et al., 2020]. Esse fenômeno acontece no aprendizado dos seres humanos. Por exemplo, assumindo que a primeira tarefa é aprender a andar de bicicleta. Em seguida se aprende a andar de moto. O fato do ser humano melhorar as habilidades em andar de moto, geralmente melhoram suas habilidade em andar de bicicleta.

Uma matriz de passos incrementais por tarefas pode ser usada para apresentar a avaliação de um modelo [Sodhani et al., 2020] de forma mais detalhada. A Tabela 3.1 mostra um exemplo desta matriz considerando T tarefas e N passos incrementais, com $N = T$. Cada linha apresenta a métrica de precisão do modelo em todas as tarefas consideradas no experimento. Cada coluna representa é o resultado de uma tarefa específica.

Passos	**Tarefa 1**	**Tarefa 2**	**Tarefa 3**	**Tarefa 4**	**...**	**Tarefa T**
Passo 1	0.99	0.15	0.07	0.17	...	0.12
Passo 2	0.90	0.98	0.10	0.16	...	0.12
Passo 3	0.80	0.82	0.97	0.22	...	0.24
Passo 4	0.76	0.75	0.83	0.98	...	0.26
			...			
Passo N	0.75	0.72	0.81	0.85	...	0.99

Tabela 3.1. A matriz de passos incrementais por tarefas pode ser utilizada para apresentar a avaliação de um modelo. A célula azul em cada linha mostra a métrica na tarefa aprendida na etapa que a linha representa. As células cinzas em cada linha mostram a métrica das tarefas já aprendidas, e o modelo visa preservar os valores arquivados em etapas antigas. Por fim, as células vermelhas em cada linha mostram a métrica em tarefas que o modelo ainda não teve acesso em nenhuma fase de treinamento. Isso avalia se o conhecimento adquirido pelo modelo até a tarefa atual trará benefícios no aprendizado futuro.

Diferentes protocolos de avaliação são adotados na literatura para avaliar os métodos que buscam mitigar o esquecimento catastrófico. Nessa seção apresentamos o mais usado na literatura. Esse protocolo é conhecido como T-Tarefas. A Figura 3.18 demonstra esse

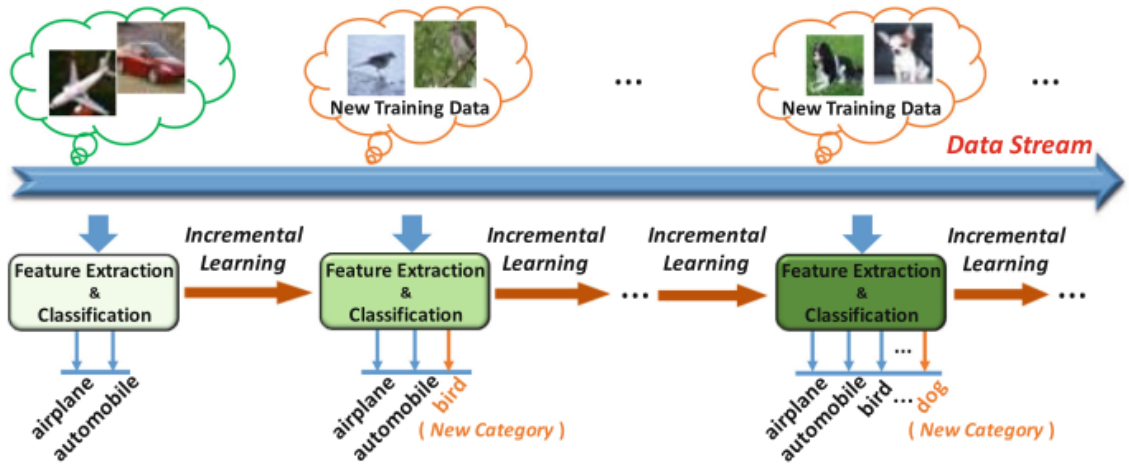


Figura 3.18. Protocolo de avaliação de N-Tarefas. Figura extraída do *survey* de Aleixo et al. [2024].

protocolo. Em cada tarefa o modelo aprende novas informações, e em seguida é avaliada a qualidade em todas as tarefas já vistas. Não existe uma métrica única na literatura que seja utilizada, entretanto é comum o uso da acurácia. Cada tarefa pode ser uma base de dados ou a mesma base dividida em T partes disjuntas.

A depender da configuração em que o modelo esta sendo avaliado, esse protocolo tem alguns ajustes. Por exemplo, no *Domain Incremental Learning* cada tarefa são amostras do mesmo problema em um domínio diferente. Já no *Task Incremental Learning*, a identificação da tarefa sempre é informada ao modelo junto com as amostras.

3.4 Trabalhos Relacionados

A proposta desta tese, apresentada no Capítulo 4, enquadra-se na categoria de **Redes Dinâmicas** para aprendizado contínuo. Para garantir uma comparação justa e coerente, concentramos nossa análise exclusivamente em métodos que adotam estratégias dinâmicas para mitigar o esquecimento catastrófico, ou seja, abordagens que expandem a arquitetura do modelo conforme novas tarefas são aprendidas.

Nesta seção, analisamos os seguintes métodos pertencentes à categoria de redes dinâmicas:

- **SeNA-CNN** [Zacarias & Alexandre, 2018b]: Inspirado no Stepwise Pathnet [Imai & Nobuhara, 2018], que demonstrou que as camadas iniciais de uma CNN extraem características de baixo nível, o SeNA-CNN replica as camadas a partir da terceira, mantendo as duas primeiras compartilhadas entre todas as tarefas. Além disso, uma CNN

auxiliar equipada com uma camada *softmax* é utilizada para identificar qual ramo (ou sub-rede) ativar durante a inferência.

- **Stepwise Pathnet** [Imai & Nobuhara, 2018]: Este trabalho fundamenta a ideia de que as camadas iniciais de uma CNN extraem características genéricas, servindo de base para estratégias dinâmicas. Embora seu foco seja a seleção de caminhos para aprendizado incremental, seus achados foram utilizados no desenvolvimento do SeNA-CNN.

- **Progressive Neural Networks (PNN)** [Rusu et al., 2016]: Para cada nova tarefa, o PNN replica toda a rede, congelando os pesos já treinados e criando conexões laterais para permitir a transferência de conhecimento. Embora eficaz na mitigação do esquecimento catastrófico, o PNN sofre com um crescimento exponencial no número de parâmetros.

- **Dynamically Expandable Representation (DER)** [Yan et al., 2021]: Esta abordagem expande o extrator de características para cada nova tarefa, mantendo um único classificador. Assim, busca adaptar a representação do modelo sem interferir excessivamente no conhecimento prévio.

- **Dynamically Expandable Network (DEN)** [Yoon et al., 2018]: O DEN adiciona dinamicamente novos neurônios e/ou camadas quando necessário, utilizando regularização para promover a esparsidade e mitigar a deriva semântica dos pesos.

- **Expert Gate** [Aljundi et al., 2017]: Essa abordagem aloca especialistas para cada tarefa e emprega mecanismos de portões para selecionar o especialista adequado durante a inferência, evitando a interferência entre tarefas.

A Tabela 3.2 apresenta uma comparação das principais características dessas abordagens, considerando se utilizam CNNs, se mitigam o esquecimento catastrófico (CF), a escalabilidade em relação ao número de tarefas e o tipo de aprendizado incremental em que são aplicadas. As Figuras 3.19 e 3.20 ilustram, respectivamente, as arquiteturas do PNN e do SeNA-CNN.

Embora todos os métodos apresentados busquem mitigar o esquecimento catastrófico por meio da expansão incremental da rede, suas abordagens diferem significativamente:

- **PNN** replica toda a rede para cada nova tarefa, garantindo a preservação do conhecimento prévio, mas resultando em um alto custo computacional e um crescimento descontrolado do modelo.

Tabela 3.2. Comparação dos métodos de Redes Dinâmicas para aprendizado contínuo. Em negrito destaca-se o método escolhido como *baseline*.

Método	Utiliza CNN	Elimina CF	Configuração	Incremento por Tarefa
SeNA-CNN	Sim	Sim	Task Incremental	Replica a metade final do modelo para cada nova tarefa.
Stepwise Pathnet	Sim	Sim	Task Incremental	Armazena e reutiliza camadas anteriores, replicando novas conforme necessário.
PNN	Não	Sim	Task Incremental	Expande o modelo adicionando novos módulos interconectados em cada camada.
DER	Sim	Não	Task Incremental	Replica integralmente o extrator de características para cada nova tarefa.
DEN	Não	Sim	Task Incremental	Adiciona neurônios progressivamente até atingir desempenho satisfatório.
Expert Gate	Sim	Não	Task Incremental	Cria um novo modelo especializado para cada tarefa.

- **DER** e **DEN** adotam uma abordagem mais controlada, expandindo dinamicamente o extrator de características e os neurônios conforme necessário, buscando um equilíbrio entre adaptabilidade e eficiência computacional.

- **Expert Gate** aloca especialistas para cada tarefa e emprega mecanismos de seleção para evitar interferências entre representações aprendidas, mas exige módulos adicionais para o roteamento adequado das inferências.

Dentre esses métodos, o **SeNA-CNN** destaca-se como o mais adequado para comparação com a proposta do Capítulo 4, pois, juntamente com o Stepwise Pathnet, é o único que efetivamente resolve o problema do esquecimento catastrófico. No entanto, como o SeNA-CNN representa uma evolução direta do Stepwise Pathnet, não faz sentido incluí-lo na comparação. Dessa forma, a análise se concentrará no SeNA-CNN, garantindo uma avaliação mais objetiva e relevante para os desafios abordados nesta tese.

3.5 Considerações Finais do Capítulo

Neste capítulo apresentamos uma introdução sobre redes neurais e como elas funcionam. Além disso, explicamos a razão pela qual esses modelos são tão suscetíveis ao problema do esquecimento catastrófico quando operamos em um ambiente de aprendizado contínuo. Em seguida apresentamos os fundamentos para compreender o campo de pesquisa de esquecimento catastrófico em redes neurais profundas, que é o objeto de pesquisa dessa tese.

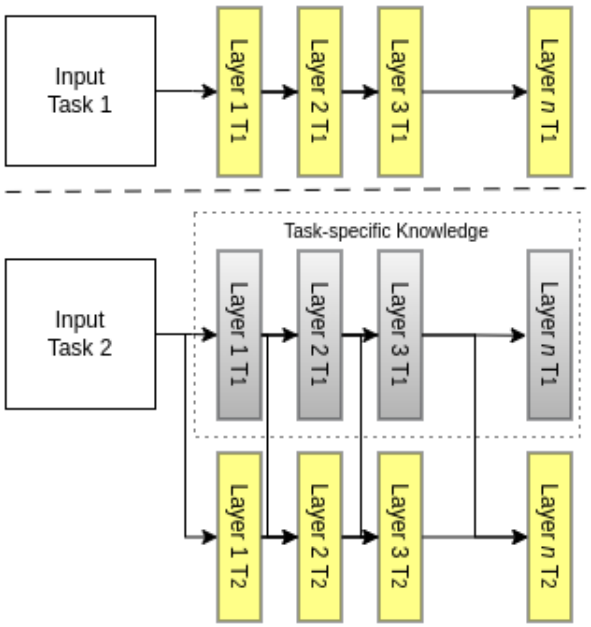


Figura 3.19. Arquitetura das Progressive Neural Networks adaptada de Rusu et al. [2016]. A cada nova tarefa, toda a estrutura é replicada e são criadas conexões entre as camadas correspondentes de redes anteriores e da rede nova. As estruturas que já passaram pelo processo de aprendizagem têm todos os seus pesos congelados para evitar o esquecimento catastrófico nas tarefas antigas.

Discutimos uma taxonomia que desenvolvemos ao longo da pesquisa para categorizar os métodos existentes na literatura.

Observamos que é possível desenvolver métodos híbridos que apresentam características de mais de uma categoria. Muitos trabalhos fazem uso de alguma técnica de *rehearsal* em conjunto com as demais para melhorar a acurácia do modelo. Entretanto, todo método que se apoia nessas técnicas violam a restrição de privacidade dos dados, não sendo viáveis em diversas aplicações do cotidiano.

Mostramos que métodos das categorias de sub-redes e os baseados em distância tendem, em determinado momento, a não conseguir mais aprender novas tarefas. No primeiro caso, isso ocorre porque os recursos do modelo (como conexões e parâmetros disponíveis) eventualmente se esgotam. Já no segundo, porque as classes das tarefas começam a apresentar sobreposição no espaço de representações, dificultando a separação entre elas.

Assim, temos evidências de que o melhor caminho a ser seguido são as redes dinâmicas, onde o modelo pode ser expandido para suportar novas tarefas. Mostramos que esse crescimento deve ser feito com cuidado, pois o modelo não pode crescer de forma proporcional a quantidade de tarefas como feito no PNN [Rusu et al., 2016].

Apresentamos, também, os seis cenários (ou configurações) nos quais um modelo pode ser aplicado e avaliado em relação ao esquecimento catastrófico. Por fim, discutimos as métricas, bem como o principal protocolo de avaliação utilizado para comparar os trabalhos

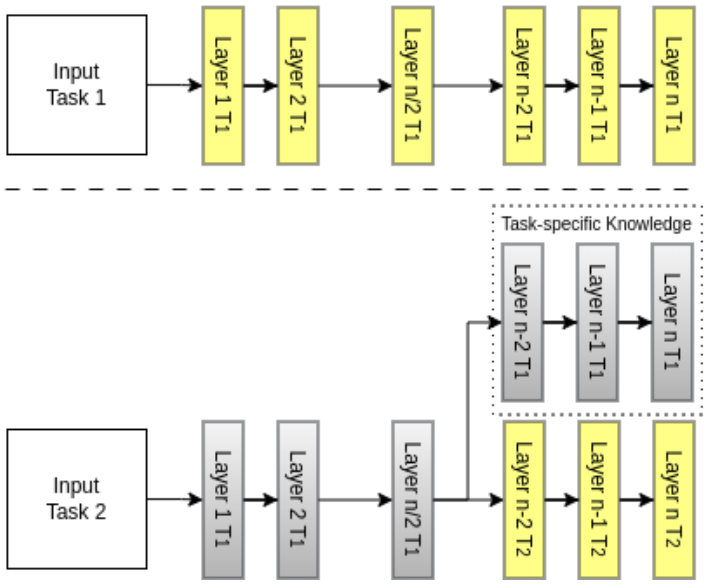


Figura 3.20. Arquitetura do SeNA-CNN. Apenas as camadas superiores são replicadas para novas tarefas, enquanto as camadas iniciais permanecem compartilhadas. Um módulo auxiliar seleciona o ramo adequado para cada inferência.

nessa área. Esse embasamento teórico é fundamental para compreender o Capítulo 4, no qual apresentamos nossa proposta utilizando adaptadores nas RNC.

4

Low-Rank Adapters para Redes Profundas Convolucionais

Low-Rank Adapters (LoRA) emergiram como uma abordagem eficiente para ajustar redes neurais profundas sem necessidade de modificar todos os seus parâmetros, reduzindo significativamente o custo computacional e o consumo de memória. Essa técnica baseia-se na fatoração de matrizes de baixa ordem, permitindo ajustes localizados e eficientes. Nos últimos anos, o LoRA tem sido amplamente empregado na adaptação de grandes modelos de linguagem (LLMs), como GPT e BERT, tornando o *fine-tuning* mais acessível e menos oneroso.

Nesta tese, exploramos o uso do LoRA e do ConvLoRA—uma adaptação do LoRA para camadas convolucionais—como uma estratégia combinada para mitigar o problema do esquecimento catastrófico (CF) em aprendizado contínuo. Essa abordagem permite adaptar modelos com um incremento mínimo de parâmetros, viabilizando a classificação eficiente de imagens em cenários de aprendizado contínuo. Devido à adição progressiva de novos parâmetros a cada tarefa, nosso *framework* se insere na categoria de Redes Dinâmicas. Ao longo deste capítulo, detalhamos os fundamentos dessa abordagem e sua implementação para lidar com CF de forma eficiente.

4.1 Introdução

Sistemas inteligentes baseados em RNC possuem uma grande capacidade de reconhecimento de imagens. Em algumas tarefas, como classificação de objetos, chegam a ultrapassar a assertividade do ser humano [Yu et al., 2017]. Para atingir esse nível de qualidade os modelos precisam de muitos dados e tempo de treinamento. Além disso, essa assertividade só ocorre

quando o modelo é treinado com amostras que apresentam a mesma distribuição estatística das amostras encontradas no ambiente de produção.

Quando uma RNC é implantada em um ambiente de produção, como em um veículo autônomo, novas classes de dados podem surgir ao longo do tempo, exigindo que o modelo expanda seu conhecimento sem perder o que já foi aprendido. No contexto do aprendizado incremental baseado em tarefas (*Task Incremental Learning*), cada nova tarefa introduz um conjunto distinto de classes que não estavam presentes no treinamento inicial. O desafio central não é apenas adaptar o modelo, mas garantir que ele consiga incorporar essas novas informações sem comprometer o desempenho nas classes previamente aprendidas. Abordagens ingênuas podem permitir que o modelo aprenda novas classes, porém, sem mecanismos adequados, esse aprendizado ocorre à custa do esquecimento do conhecimento anterior. No aprendizado incremental, é essencial que o modelo agregue conhecimento progressivamente, preservando as classes já aprendidas em vez de simplesmente substituí-las.

Considere, por exemplo, um sistema de veículo autônomo treinado para reconhecer um subconjunto específico de placas de trânsito, adequado para operar em uma determinada região. Ao ser implantado em uma nova localidade, onde o conjunto de placas de trânsito é mais amplo e inclui sinalizações não vistas anteriormente, o modelo precisaria ser atualizado para reconhecer essas novas classes sem comprometer sua capacidade de identificar as placas já aprendidas. Essa adaptação envolve dois desafios principais: i) garantir que o conhecimento adquirido sobre as placas previamente aprendidas não seja perdido, evitando o problema do esquecimento catastrófico [McCloskey & Cohen, 1989]; e ii) evitar a necessidade de um novo treinamento completo utilizando todas as placas de trânsito conhecidas, pois isso seria computacionalmente inviável.

Embora uma RNC tenha a capacidade de classificar imagens de gatos e cachorros com alta precisão, é necessário que ela tenha acesso simultâneo a amostras de ambas as classes durante o treinamento para garantir um bom desempenho. No entanto, isso não ocorre no cenário de aprendizado incremental, pois novas classes de imagens vão surgindo ao longo do tempo, quando o modelo já está em produção. Se inicialmente o modelo for treinado apenas com imagens de gatos e, posteriormente, precisar aprender a classificar cachorros, a falta de acesso às imagens anteriores pode levar ao esquecimento catastrófico, fazendo com que ele perca a capacidade de reconhecer gatos corretamente.

Isso caracteriza a configuração de *task incremental learning* (Seção 3.2). O principal objetivo do ConvLoRA é permitir que os modelos RNC aprendam novas tarefas de maneira incremental sem comprometer a performance em tarefas anteriores e sem a necessidade de armazenamento de dados antigos. Isso é alcançado através da introdução de matrizes de adaptação de baixa dimensionalidade que são adicionadas às camadas convolucionais existentes, conforme detalhado nas subseções a seguir.

4.2 Adaptadores

O treinamento de RNC geralmente exige a atualização de todos os pesos do modelo, o que é computacionalmente custoso e consome grandes quantidades de memória. Uma alternativa comum é o ajuste fino das últimas camadas, mas essa abordagem nem sempre atinge a precisão desejada, pois limita a capacidade do modelo de se adaptar completamente a novas tarefas. Para contornar essas limitações, surgiram os adaptadores, que permitem modificar modelos pré-treinados ajustando apenas um subconjunto reduzido de parâmetros. Essa estratégia equilibra eficiência e desempenho, reduzindo o custo computacional sem comprometer a qualidade da adaptação.

Como os parâmetros pré-treinados não são alterados durante o treinamento de uma nova tarefa, nosso *framework* garante que o modelo não sofrerá de esquecimento catastrófico. Em vez de modificar diretamente os pesos originais da rede, os adaptadores introduzem componentes adicionais responsáveis por capturar as novas informações sem interferir no conhecimento previamente adquirido. A seguir vamos detalhar a técnica dos adaptadores de baixa ordem em camadas completamente conectados (onde foram inicialmente propostos).

4.2.1 LoRA

A *Low-Rank Adaptation* (LoRA) é uma técnica desenvolvida para reduzir significativamente o número de parâmetros treináveis em grandes modelos de linguagem (LLMs) [Brown et al., 2020, Liu et al., 2020b]. Essa abordagem introduz um conjunto reduzido de novos pesos ao modelo, garantindo que apenas esses parâmetros adicionais sejam ajustados durante o treinamento. Os adaptadores do LoRA são inseridos em todas as camadas totalmente conectadas, tornando o processo de ajuste fino mais eficiente em termos de recursos computacionais, especialmente para modelos de grande escala [Hu et al., 2022].

O método LoRA funciona congelando a matriz de pesos pré-treinados da camada, denotada por $W \in \mathbb{R}^{d\times d}$, e adicionando duas matrizes menores, A e B, que são as únicas atualizadas durante o treinamento. Apesar do tamanho reduzido de A e B em comparação com W, o produto matricial AB gera uma nova matriz com as mesmas dimensões de W, permitindo a adição elemento a elemento:

$$W' = W + AB \tag{4.1}$$

Isso possibilita que o modelo aprenda novos padrões sem modificar diretamente os pesos originais, preservando assim o conhecimento previamente adquirido. Além disso, o LoRA pode ser estendido para tensores de qualquer dimensão, o que amplia sua aplicabilidade para diferentes tipos de redes neurais.

A Figura 4.1 ilustra o funcionamento do LoRA. O método consiste em congelar os pesos pré-treinados da camada e introduzir dois vetores treináveis, A e B. O vetor A é inicializado com uma distribuição normal $\mathcal{N} \sim (0, \sigma^2)$, enquanto B é inicializado com valores nulos. Esses vetores são consideravelmente menores que W, tornando o ajuste fino mais eficiente.

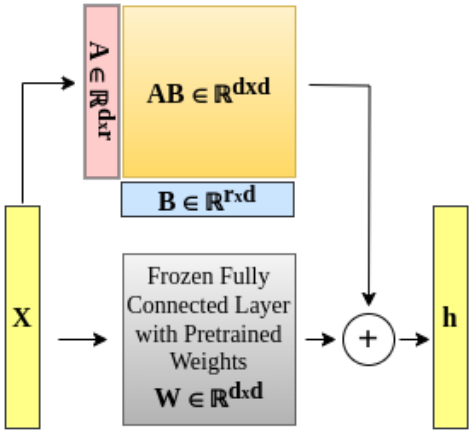


Figura 4.1. O método LoRA congela os pesos pré-treinados W de uma camada e aprende dois vetores adicionais, A e B, reduzindo significativamente o número de parâmetros treináveis.

4.2.2 ConvLoRA

A arquitetura do ConvLoRA é fundamentada na modificação das camadas convolucionais das através da integração de matrizes de adaptação de baixa dimensionalidade. Esta abordagem é inspirada na técnica LoRA, que tem sido utilizada com sucesso em modelos baseados em *transformers*.

Na arquitetura padrão de uma RNC, as camadas convolucionais são parametrizadas por um tensor de pesos $W \in \mathbb{R}^{d \times d \times c_{\text{in}} \times c_{\text{out}}}$, onde d representa a dimensão espacial do *kernel*, c_{in} é o número de canais de entrada e c_{out} corresponde ao número de filtros (ou canais de saída). Cada um dos c_{out} filtros convolucionais contém c_{in} matrizes de dimensão $d \times d$, resultando em um total de $|W| = c_{\text{in}} \cdot c_{\text{out}} \cdot d \cdot d$ parâmetros treináveis por camada convolucional.

Um adaptador ConvLoRA pode ser implementado de diversas maneiras. Propomos uma implementação baseada na decomposição dos filtros convolucionais em três matrizes A, B e C, fatorizando a estrutura dos pesos da convolução. Essa estratégia tem como objetivo reduzir a complexidade do modelo sem comprometer sua expressividade. Para garantir a compatibilidade com múltiplos canais de entrada e saída, as dimensões dessas matrizes foram definidas como:

$$A \in \mathbb{R}^{1 \times c_{\text{in}} \times d \times 1}, \quad B \in \mathbb{R}^{1 \times c_{\text{in}} \times 1 \times d}, \quad C \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times 1 \times 1}.$$

Nessa decomposição, as matrizes A e B capturam a estrutura espacial do *kernel* ao longo das dimensões convolucionais, enquanto C atua como um ajuste específico para os canais, aprendendo a interação entre os filtros de entrada e saída. Essa abordagem permite uma redução significativa no número de parâmetros treináveis, mantendo a capacidade da camada convolucional de extrair representações discriminativas dos dados.

Essas matrizes são combinadas para ajustar os pesos dos filtros de maneira eficiente. A adaptação é realizada através da combinação elementar dessas matrizes com os filtros originais:

$$\hat{W} = W + \alpha(A \cdot B \cdot C)$$

Onde α é um fator de escala que controla a contribuição das novas matrizes de adaptação. Essa formulação permite que o modelo aprenda novos padrões complexos com um aumento mínimo no número de parâmetros.

Para ilustrar, considere um exemplo onde uma camada convolucional possui 32 filtros, cada um com um tamanho de *kernel* de 7x7 que irá processar uma imagem RGB (c_in=3). Isso resulta em um total de 4704 parâmetros de *kernels* mais 32 parâmetros de *bias*, totalizando 4736 parâmetros. Aplicando essa implementação do ConvLoRA, podemos criar um adaptador dessa camada com as três matrizes de adaptação A, B e C, cada uma com dimensões significativamente menores. Por exemplo, A teria dimensão 1x3x7x1, B seria 1x3x7x1 e C seria 32x3x1x1. Isso resulta em apenas 138 parâmetros, uma redução de aproximadamente 97% em comparação com a camada convolucional original. A Figura 4.2 ilustra essa implementação.

Embora o exemplo acima sugira que a implementação de um adaptador para uma camada convolucional reduz significativamente o número de parâmetros, essa redução pode ser menos expressiva em camadas que possuem um grande número de filtros e operam sobre um alto número de *feature maps*, devido ao crescimento do tensor C. Essa característica é especialmente relevante nas camadas mais profundas das RNC modernas.

Por exemplo, na arquitetura VGG-19, algumas camadas convolucionais possuem 512 *feature maps* de entrada e geram 512 *feature maps* de saída, utilizando *kernels* de dimensão 3×3. Nesse cenário, a implementação proposta geraria um adaptador com 265.216 parâmetros, em contraste com os 2.359.808 da camada convolucional original, representando uma redução aproximada de 89%.

Concomitantemente à nossa proposta, Aleem et al. [2024] propuseram uma abordagem alternativa de implementação, menos sensível à variação na quantidade de *feature maps*. Nesta abordagem, são utilizados os mesmos tensores A e B da técnica LoRA. No entanto, após a multiplicação desses tensores, a matriz resultante é redimensionada para possuir a

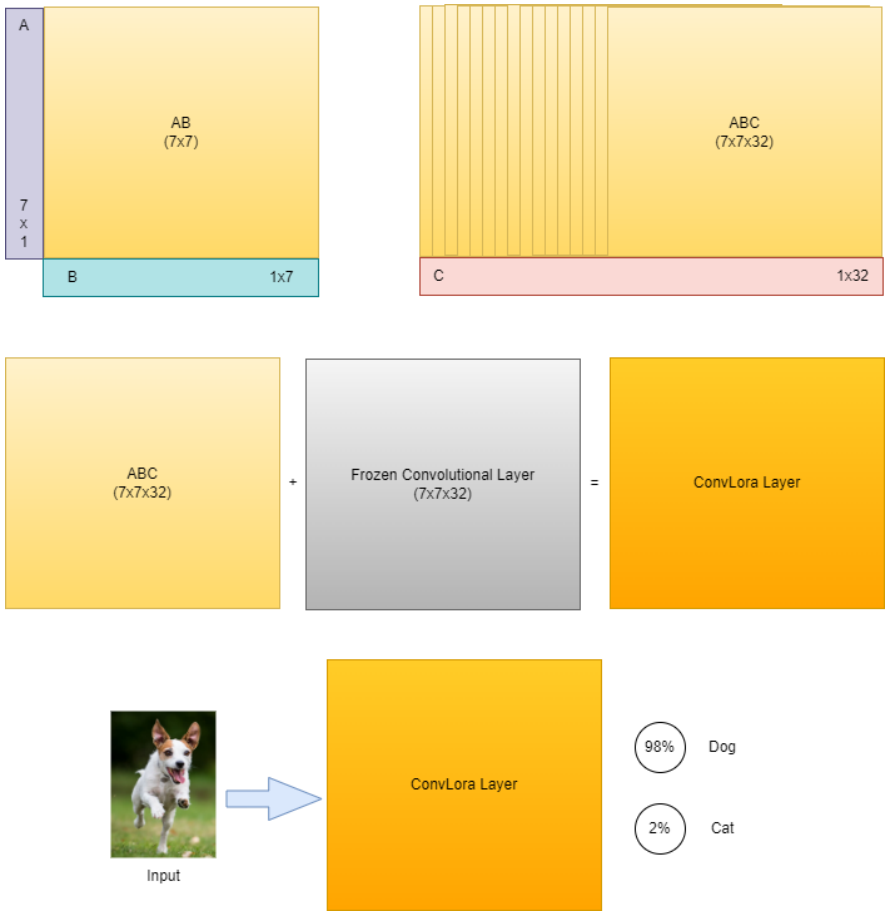


Figura 4.2. Na parte superior da figura, podemos ver a formação dos parâmetros treináveis utilizando as matrizes A, B e C, aos quais chamamos de núcleos efêmeros. No meio, é possível observar os núcleos efêmeros sendo somados com pesos congelados, gerando a camada ConvLora. Na parte inferior, é apresentada a utilização tanto no treinamento quanto na predição. A camada ConvLora pode substituir uma camada convolucional em qualquer arquitetura, como VGG ou ResNet.

mesma forma dos filtros convolucionais, conforme ilustrado na Figura 4.3.

Para que a matriz resultante da multiplicação dos tensores A e B possa ser transformada na mesma dimensionalidade dos filtros W, esses tensores devem possuir as dimensões $A \in \mathbb{R}^{d\times m}$, $B \in \mathbb{R}^{m\times k}$, onde d, m e k são calculados a partir do hiperparâmetro r (já conhecido do LoRA) em conjunto com as configurações da camada que será adaptada. Esses valores são definidos como: $d = c_{\text{in}} \times kernel\_size$, $k = c_{\text{out}} \times kernel\_size$, $m = r \times kernel\_size$. Dessa forma, d representa o número total de elementos de entrada considerados na operação convolucional, sendo determinado pelo número de canais de entrada c_{in} e pelo tamanho do *kernel*. O valor de k define a quantidade de elementos de saída gerados, diretamente proporcional ao número de canais de saída c_{out} e ao tamanho do *kernel*. Por fim, m atua como um fator de redução intermediário, influenciado pelo hiperparâmetro r e pelo tamanho do *kernel*, controlando a dimensão do espaço latente onde a transformação ocorre.

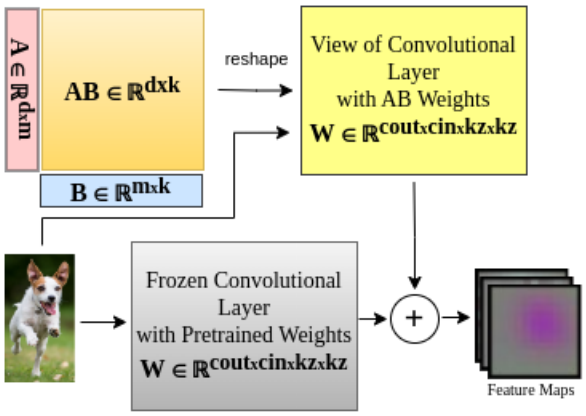


Figura 4.3. A matriz ConvLoRA é gerada pelo produto interno de dois pequenos vetores treináveis, A e B, e posteriormente redimensionada para possuir as mesmas dimensões de W. As inicializações seguem o mesmo padrão do LoRA. Os valores d, m e k são calculados com base nos hiperparâmetros da camada, enquanto r é definido pelo arquiteto da rede.

Para ilustrar essa redução, considere o cenário discutido anteriormente em que $c_{\text{in}} = c_{\text{out}} = 512$ e o *kernel* é de tamanho 3. Em uma camada convolucional, o número total de parâmetros dos filtros seria: $512 \times 512 \times 3 \times 3 = 2.359.296$. Nessa implementação do `ConvLoRA`, os tensores A e B introduzem um novo conjunto de parâmetros. Se definirmos $r = 16$, então temos: $d = 512 \times 3 = 1536, \quad m = 16 \times 3 = 48, \quad k = 512 \times 3 = 1536$. Assim, o tensor A terá dimensões 1536×48 e o tensor B terá dimensões 48×1536, e o número total de novos parâmetros é dado por:

$$\underbrace{1536 \times 48}_{\text{parâmetros de } A} + \underbrace{48 \times 1536}_{\text{parâmetros de } B} = 2 \times (1536 \times 48) = 147.456.$$

Comparando com os 2.359.296 parâmetros originais, temos que o ConvLoRA, com $r = 16$, utiliza apenas 147.456 parâmetros treináveis, o que corresponde a uma redução de aproximadamente $1 - {}^{147.456}/_{2.359.296} \approx 0,9375$ ou **93,7%** na quantidade de parâmetros. Essa significativa diminuição reduz os custos computacionais e de memória durante o treinamento, mantendo a capacidade da camada de aprender novas características.

Para os resultados apresentados nesta tese, utilizamos essa implementação, pois, apesar de ambas as abordagens apresentarem desempenhos semelhantes em termos de acurácia, a segunda abordagem demonstra uma maior eficiência na economia de parâmetros. É importante ressaltar que o principal objetivo da aplicação de adaptadores nos modelos para mitigar o problema do esquecimento catastrófico é justamente garantir um incremento mínimo de parâmetros treináveis a cada nova tarefa, preservando a eficiência do modelo.

A técnica ConvLoRA não adapta as camadas de Normalização por Lote (*Batch Normalization*), pois essas camadas armazenam estatísticas específicas das instâncias da base de

dados utilizadas no treinamento de cada tarefa. Para assegurar a correta normalização das ativações ao longo do aprendizado contínuo, cada nova tarefa recebe uma cópia independente dessas camadas, preservando as versões correspondentes às tarefas anteriores. Como as camadas de *Batch Normalization* representam uma fração insignificante do total de parâmetros da rede (por exemplo, em uma VGG-19, essas camadas possuem aproximadamente 11 mil parâmetros, enquanto a rede completa contém 139 milhões), essa estratégia não impõe um aumento expressivo na complexidade computacional ou na memória requerida para o treinamento do modelo.

4.2.2.1 Funcionamento do ConvLoRA

O funcionamento do ConvLoRA envolve a introdução de novas matrizes de adaptação durante o treinamento das CNNs. O processo pode ser descrito em três etapas principais:

Congelamento dos Pesos Originais: Durante o treinamento inicial, os pesos originais W das camadas convolucionais são treinados normalmente. Após o treinamento inicial, esses pesos são congelados e não são atualizados durante a adaptação a novas tarefas. Isso preserva o conhecimento das tarefas anteriores e evita o CF.

Por exemplo, considere um modelo treinado para classificar imagens de gatos e cachorros. Após o treinamento inicial, os pesos que permitem essa classificação são congelados. Quando novas classes, como pássaros, são adicionadas, os pesos originais não são alterados.

Treinamento das Matrizes de Adaptação: Para cada nova tarefa, apenas as matrizes de adaptação A e B são treinadas. Isso é feito mantendo os pesos originais congelados, o que permite a adaptação eficiente do modelo às novas tarefas sem comprometer o desempenho nas tarefas anteriores.

Seguindo o exemplo anterior, quando imagens de pássaros são introduzidas, apenas as novas matrizes A e B são ajustadas para acomodar essa nova classe, preservando a capacidade do modelo de classificar gatos e cachorros.

Combinação dos Pesos: Durante a inferência, os novos filtros convolucionais $\hat{W}$ são obtidos combinando os pesos originais com as matrizes de adaptação. Isso permite que o modelo mantenha a precisão nas tarefas anteriores enquanto se adapta a novas tarefas.

Continuando com o mesmo exemplo, durante a inferência, a combinação dos pesos originais com as matrizes de adaptação permite que o modelo classifique corretamente tanto as novas imagens de pássaros quanto as antigas de gatos e cachorros.

Estas etapas são ilustradas na Figura 4.3, onde o ConvLoRA é mostrado como uma camada adicional que combina os *kernels* efêmeros treináveis com os pesos congelados. Essa camada pode ser facilmente integrada em diversas arquiteturas de redes convolucionais, como VGG ou ResNet, proporcionando flexibilidade e eficiência no treinamento contínuo.

4.3 Vantagens dos Adaptadores

Os adaptadores apresentam diversas vantagens em comparação com os métodos tradicionais de mitigação do *Catastrophic Forgetting* (CF), tornando-se uma solução promissora para aprendizado contínuo em redes neurais convolucionais (RNC).

Redução do Aumento de Parâmetros: O ConvLoRA minimiza significativamente o crescimento no número de parâmetros necessários para a adaptação a novas tarefas. Especificamente, a implementação do ConvLoRA resulta em um acréscimo de aproximadamente 6% no número de parâmetros das camadas convolucionais, quando comparado ao modelo original VGG19. Essa característica permite que novos conhecimentos sejam incorporados ao modelo sem que haja um crescimento descontrolado na complexidade do sistema.

Eliminação da Necessidade de Armazenamento de Dados Anteriores: Diferentemente das abordagens que requerem a retenção de dados históricos para evitar a degradação do desempenho em tarefas anteriores, os adaptadores eliminam essa necessidade. Essa propriedade é particularmente vantajosa em cenários onde a privacidade e a segurança dos dados são críticas, como na área médica. Nesses casos, a possibilidade de aprendizado incremental sem necessidade de armazenamento de dados prévios representa um avanço significativo na viabilidade prática de sistemas de aprendizado contínuo.

Preservação da Precisão: Os adaptadores demonstram capacidade de preservar a precisão das tarefas previamente aprendidas, mitigando os efeitos do CF. Resultados experimentais apresentados no Capítulo 5 indicam que essa abordagem mantém tanto a precisão inicial quanto a final em *benchmarks* amplamente utilizados no aprendizado contínuo, como CIFAR-100 e CUB-200. Essa preservação da acurácia confere maior estabilidade ao modelo ao longo do tempo, garantindo um desempenho consistente mesmo após múltiplas fases de aprendizado.

Eficiência Computacional: A eficiência computacional dos adaptadores destaca-se pelo controle no aumento do número de parâmetros, o que se traduz em menores exigências de memória. Essa característica torna os adaptadores uma alternativa viável para cenários onde os recursos computacionais são restritos. Em ambientes corporativos, por exemplo, onde a infraestrutura de hardware pode ser limitada, o ConvLoRA viabiliza a implementação de modelos de aprendizado contínuo sem a necessidade de investimentos substanciais em novos equipamentos.

Dessa forma, os adaptadores oferecem uma solução robusta e eficiente para o problema do *Catastrophic Forgetting*, permitindo que modelos de RNC se adaptem continuamente a novas tarefas sem comprometer o desempenho em tarefas anteriores, ao mesmo tempo em que mantêm a eficiência computacional. Esses fatores reforçam a relevância dos adaptadores como um mecanismo essencial para o avanço do aprendizado contínuo em redes neurais

convolucionais.

4.4 Considerações Finais

Neste capítulo, exploramos o conceito de *Low-Rank Adapters* (LoRA) e sua adaptação para redes convolucionais, o ConvLoRA. Discutimos como essas técnicas permitem a adaptação eficiente de RNP, minimizando o número de parâmetros treináveis e, consequentemente, reduzindo o custo computacional e a necessidade de armazenamento de dados anteriores. A abordagem proposta se enquadra na categoria de Redes Dinâmicas, pois permite que novos parâmetros sejam incorporados ao modelo para aprender uma nova tarefa.

Ao longo do capítulo, apresentamos os fundamentos teóricos do LoRA e sua aplicação em modelos de aprendizado contínuo. Explicamos como o ConvLoRA adapta as camadas convolucionais através da introdução de matrizes de baixa dimensionalidade, garantindo que a rede aprenda novas tarefas sem sofrer com o problema do *Catastrophic Forgetting*. Além disso, destacamos as vantagens dessa técnica em relação a abordagens tradicionais, como a eficiência computacional e a eliminação da necessidade de armazenamento de dados antigos.

No próximo capítulo, apresentamos os resultados experimentais obtidos com a aplicação dos adaptadores em *benchmarks* de aprendizado contínuo. Avaliamos seu impacto na preservação da acurácia ao longo das tarefas e comparamos seu desempenho com técnicas alternativas. Os experimentos reforçam a eficácia do método proposto, demonstrando sua viabilidade para cenários reais de aprendizado contínuo.

5

Resultados

Neste capítulo, apresentamos os resultados obtidos através da aplicação de adaptadores em redes neurais convolucionais para superar o problema do esquecimento catastrófico (CF) em tarefas contínuas de classificação de imagens. Os experimentos foram conduzidos utilizando os conjuntos de dados CIFAR-100 e CUB-200, comparando com a abordagem baseline SenaCNN [Zacarias & Alexandre, 2018b].

5.1 Metodologia

Os experimentos foram realizados utilizando a arquitetura VGG-19, devido à sua capacidade comprovada de aprender eficientemente ambos os conjuntos de dados, sem as complexidades adicionais introduzidas por conexões residuais. Cada tarefa foi treinada até a convergência da função de perda, utilizando a técnica de *early stopping* com paciência de 5 épocas. O otimizador Adam foi escolhido com uma taxa de aprendizado de 1×10^{-5} e um decaimento de peso de 5×10^{-4}. Os tamanhos dos lotes foram 64 para CIFAR-100 e 8 para CUB-200 devido a restrições computacionais.

5.1.1 Descrição dos Conjuntos de Dados

Para a avaliação da eficácia dos adaptadores, utilizamos dois conjuntos de dados amplamente reconhecidos na comunidade de aprendizado contínuo: CIFAR-100 e CUB-200. O conjunto de dados CIFAR-100 é composto por 100 classes, representando uma ampla variedade de objetos comuns, como animais, veículos e objetos domésticos. Essas classes são distribuídas em 20 superclasses, cada uma contendo 5 classes relacionadas. Para os experimentos de aprendizado contínuo, dividimos o CIFAR-100 em dez tarefas, cada uma contendo 10 classes. Cada imagem no CIFAR-100 tem uma resolução de 32x32 pixels e está em formato

RGB (3 canais de cor). Este conjunto de dados é desafiador devido à sua baixa resolução e à alta variabilidade dentro de cada classe. A Figura 5.1 apresnta algumas amostras dessa base de dados.

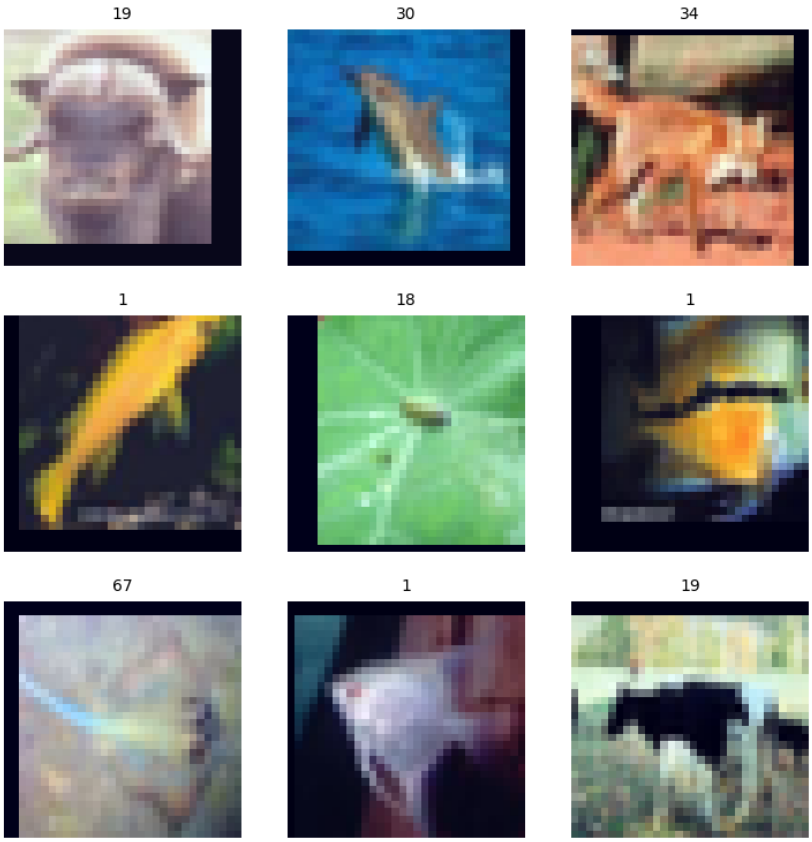


Figura 5.1. Exemplos de amostras do conjunto de dados CIFAR-100, ilustrando a diversidade de classes. O número acima de cada amostra indica a classe correspondente.

O conjunto de dados CUB-200-2011 (Caltech-UCSD Birds-200-2011) é uma coleção detalhada de imagens de aves, abrangendo 200 classes de diferentes espécies de aves. Este conjunto de dados é notavelmente mais complexo do que o CIFAR-100 devido à alta variabilidade visual entre as espécies de aves e à presença de detalhes finos nas imagens. Para os nossos experimentos, dividimos o CUB-200 em 5 tarefas: a primeira tarefa contém 100 classes, enquanto as quatro tarefas subsequentes contêm 25 classes cada. Todas as imagens foram redimensionadas para 128x128 pixels e normalizadas para garantir consistência em termos de tamanho e distribuição de cor (128x128x3). Este conjunto de dados é especialmente adequado para avaliar técnicas de aprendizado contínuo devido à sua complexidade e à necessidade de preservar detalhes finos ao longo das tarefas. A Figura 5.2 apresenta algumas amostras dessa base de dados.

Estes conjuntos de dados foram selecionados para testar a eficácia dos adaptadores em cenários de aprendizado contínuo com diferentes níveis de complexidade e variabilidade

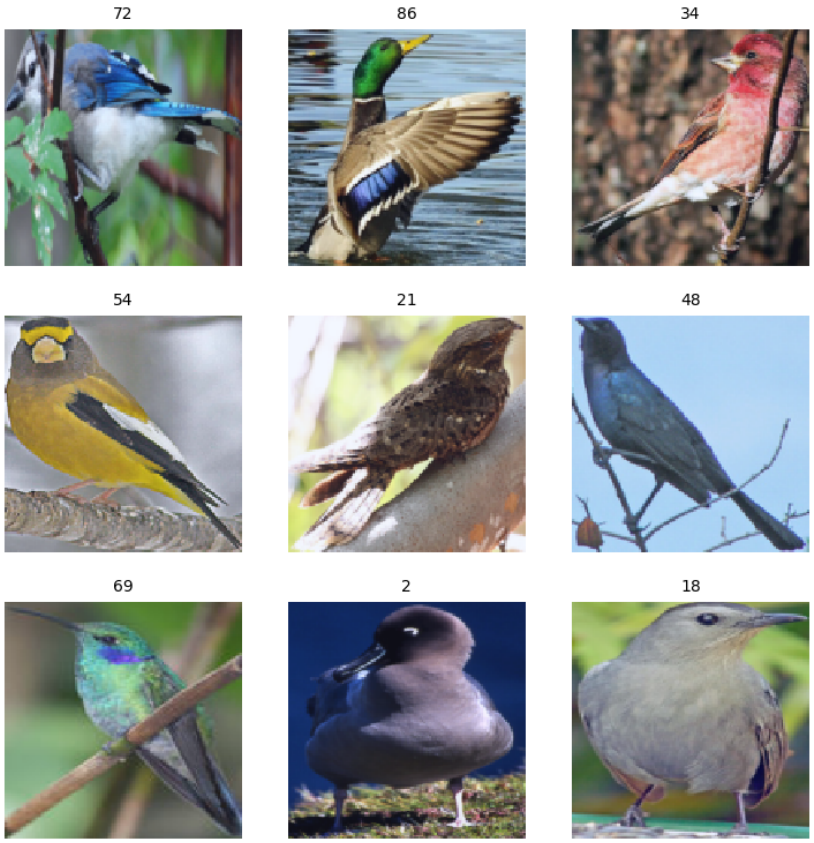


Figura 5.2. Exemplos de amostras do conjunto de dados CUB-200, representando diferentes espécies de aves. O número acima de cada amostra indica a classe correspondente.

visual.

5.1.2 Definição do Hiperparâmetro *r* para ConvLoRA

Para determinar o valor ideal do hiperparâmetro *r* no ConvLoRA, realizamos uma série de experimentos utilizando a arquitetura VGG19 e o conjunto de dados CUB-200. O objetivo foi identificar o menor valor de *r* que preservasse a acurácia do modelo próxima ao limite superior estabelecido pelo fine-tuning completo, minimizando a adição de novos parâmetros e, consequentemente, o custo computacional.

O conjunto de dados foi reduzido para 100 classes, e um modelo VGG19 pré-treinado no ImageNet foi utilizado como ponto de partida dos parametros da rede neural. A camada de classificação original foi substituída para acomodar as novas classes, resultando em uma acurácia inicial de aproximadamente 2%, compatível com a aleatoriedade.

Para compreender o impacto do ajuste fino (fine-tuning) na performance do modelo, avaliamos quatro configurações experimentais diferentes:

- **Fine-tuned Upper Bound**: todas as camadas do modelo foram treinadas, incluindo

as camadas convolucionais, totalmente conectadas e de normalização em lote. Esta configuração representa o limite superior de desempenho.

- **Finetune-1**: todas as camadas foram congeladas, exceto as camadas de normalização em lote e a camada de classificação.
- **Finetune-2**: além das camadas de normalização em lote, descongelamos as duas últimas camadas totalmente conectadas.
- **Finetune-3**: além das camadas de normalização em lote, descongelamos todas as três camadas totalmente conectadas da VGG19.

Com base nesses experimentos, adotamos a configuração **Finetune-3** como referência para a escolha de r no ConvLoRA, pois essa configuração apresentou uma maior acurácia. Para calibrar o valor de r, realizamos experimentos com os seguintes valores $r \in \{1,3,6,9,12,24\}$, onde cada configuração ConvLoRA-r foi testada de forma independente. A métrica utilizada foi a acurácia final alcançada pelo modelo. Nosso critério de escolha foi o menor valor de r que permitisse manter a acurácia próxima à obtida com a configuração **Fine-tuned Upper Bound**, garantindo um equilíbrio entre desempenho e eficiência computacional.

Após a definição de r para as camadas convolucionais, investigamos a aplicabilidade da técnica LoRA às camadas totalmente conectadas, que possuem uma alta densidade de parâmetros. Para isso, avaliamos três configurações adicionais:

- **FullyLoRA-1**: treinamos apenas os filtros ConvLoRA e a camada de classificação, inicializada aleatoriamente.
- **FullyLoRA-2**: treinamos os filtros ConvLoRA e aplicamos LoRA com $r = 30$ apenas às duas últimas camadas totalmente conectadas.
- **FullyLoRA-3**: treinamos os filtros ConvLoRA e aplicamos LoRA com $r = 30$ a todas as três camadas totalmente conectadas.

Em todos os experimentos, para mitigar o impacto das camadas de normalização em lote, adotamos a estratégia de criar cópias independentes para cada nova tarefa. Essa abordagem previne interferências entre distribuições de dados distintas e reduz os efeitos de mudanças de domínio, sem um aumento significativo no número de parâmetros.

O fluxo metodológico para a escolha do hiperparâmetro r seguiu os seguintes passos:

1. Seleção de um subconjunto de 100 classes do CUB-200 e ajuste da arquitetura VGG19 para esse conjunto;

2. Avaliação de diferentes estratégias de fine-tuning para estabelecer um limite superior de desempenho;

3. Seleção de um conjunto de valores para r e experimentação com ConvLoRA na configuração **Finetune-3**;

4. Identificação do menor valor de r que preservasse a acurácia próxima ao limite superior, minimizando a adição de novos parâmetros;

5. Avaliação da aplicação do LoRA às camadas totalmente conectadas e comparação com o valor ótimo de r encontrado para ConvLoRA; e

6. Implementação da estratégia de cópias independentes das camadas de normalização em lote para mitigar interferências no aprendizado contínuo.

Os resultados dessa análise serão apresentados na próxima seção.

5.2 Resultados e Discussão

5.2.1 Hiperparâmetro r e Limite Superior

Para determinar os melhores valores dos hiperparâmetros, utilizamos a arquitetura VGG19 com pesos pré-treinados no ImageNet como modelo de referência. Em seguida, realizamos um ajuste fino de todos os parâmetros da rede neural para adaptá-la a uma nova tarefa: a classificação de imagens em um subconjunto contendo as 100 primeiras classes do conjunto de dados CUB-200. Nessa configuração, o modelo atingiu uma acurácia de 77,3% com aproximadamente 140 milhões de parâmetros treináveis.

A Tabela 5.1 ilustra a relação entre o número total de parâmetros treináveis e a acurácia obtida. O modelo de referência, denominado **Fine-tuned Upper Bound**, no qual todas as camadas foram ajustadas, alcançou uma acurácia de 77,3% utilizando aproximadamente 140 milhões de parâmetros treináveis. Em contrapartida, os modelos que restringiram o ajuste fino apenas às camadas totalmente conectadas (**Finetune-1**, **Finetune-2** e **Finetune-3**) apresentaram acurácias inferiores. Esses resultados indicam que as abordagens tradicionais de aprendizado por transferência, que atualizam exclusivamente as camadas totalmente conectadas, são insuficientes para atingir o mesmo desempenho obtido pelo ajuste fino de todos os parâmetros do modelo.

Dessa forma, a maximização do desempenho requer a atualização de todas as camadas da rede, em vez de restringir os ajustes às camadas finais. Esse resultado reforça a importância da adaptação das camadas convolucionais no aprendizado de novas tarefas e evidencia a

Tabela 5.1. Comparação entre estratégias de aprendizado por transferência para adaptação a uma nova tarefa.

Nome	Parâmetros Treináveis	Acurácia
Fine-tuned Upper Bound	≈140.000.000	0,77
Finetune-3	120.365.256	0,61
Finetune-2	17.600.712	0,59
Finetune-1	819.400	0,58

Tabela 5.2. Configurações do ConvLoRA: número de parâmetros treináveis e acurácia.

Nome	r	Parâmetros Treináveis	Acurácia
ConvLoRA-1	1	94.491	0,65
ConvLoRA-3	3	283.473	0,74
ConvLoRA-6	6	566.946	0,73
ConvLoRA-9*	9	850.419	0,76
ConvLoRA-12	12	1.133.892	0,76
ConvLoRA-24	24	2.267.784	0,76

necessidade de métodos que permitam sua atualização eficiente, minimizando o crescimento do número de parâmetros e os custos computacionais.

À medida que compreendemos a importância do treinamento das camadas convolucionais para alcançar uma melhor acurácia em novas tarefas, nosso objetivo é adaptá-las minimizando a sobrecarga em termos de parâmetros treináveis. Para esse fim, aplicamos o ConvLoRA a todas as camadas convolucionais e avaliamos diferentes valores do hiperparâmetro r. A Tabela 5.2 apresenta os resultados para cada configuração, destacando a relação entre r, o número de parâmetros treináveis e a acurácia obtida. Esses resultados indicam que, embora o aumento de r geralmente melhore a acurácia, há um ponto de retorno decrescente, além do qual a adição de mais parâmetros não resulta em ganhos substanciais de desempenho.

A configuração **ConvLoRa-9** permitiu alcançar uma acurácia próxima à do limite superior obtido com ajuste fino completo, no qual todos os parâmetros são treinados. Essa configuração reduziu a quantidade de parâmetros treináveis de aproximadamente 20 milhões para cerca de 850.000, representando uma redução de 95,75%. No entanto, as camadas totalmente conectadas ainda contêm um número significativo de parâmetros treináveis. Dessa forma, fixamos o hiperparâmetro r em 9 e aplicamos o LoRA às camadas totalmente conectadas. Os resultados sugerem que configurações com um número maior de parâmetros treináveis não resultam em melhorias adicionais de desempenho, indicando que o ConvLoRA atinge um limite superior em termos de acurácia, concluindo-se que chegou na sua capacidade limite de aprendizado.

Tabela 5.3. Comparação da acurácia ao adicionar LoRA às camadas totalmente conectadas.

Nome	Parâmetros Treináveis Convolucionais	Parâmetros Treináveis Totalmente Conectados	Acurácia
FullyLoRA-1	850.419	819.400	0,75
FullyLoRA-2	850.419	2.069.560	0,73
FullyLoRA-3	850.419	1.940.680	0,76

A Tabela 5.3 apresenta a relação entre o número de parâmetros treináveis nas camadas totalmente conectadas e a acurácia obtida em diferentes configurações. Em todos os casos, a configuração **ConvLoRa-9** foi aplicada às camadas convolucionais, resultando em um total fixo de 850.419 parâmetros treináveis.

A configuração **FullyLoRA-1** aplica LoRA apenas à camada de classificação (terceira camada), resultando em 819.400 parâmetros treináveis. Já a configuração **FullyLoRA-2** estende o uso de LoRA para as três camadas totalmente conectadas e, adicionalmente, treina integralmente os parâmetros da terceira camada, totalizando 2.069.560 parâmetros treináveis. Por fim, a configuração **FullyLoRA-3** aplica LoRA às duas primeiras camadas totalmente conectadas, enquanto mantém todos os parâmetros da terceira camada treináveis, resultando em um total de 1.940.680 parâmetros treináveis.

Os resultados indicam que a aplicação seletiva do LoRA em camadas totalmente conectadas proporciona um compromisso eficiente entre precisão e economia de parâmetros treináveis. Notavelmente, enquanto a configuração **FullyLoRA-3** obteve a melhor acurácia (0,76), a **FullyLoRA-1**, que aplica LoRA apenas à camada de classificação, apresentou um desempenho comparável (0,75) com um número significativamente menor de parâmetros treináveis. Esses achados indicam que a adaptação das camadas convolucionais via ConvLoRA é um fator determinante no desempenho do modelo, enquanto a aplicação seletiva do LoRA em camadas totalmente conectadas específicas oferece um equilíbrio prático entre acurácia e eficiência paramétrica.

Com base nos resultados obtidos, a configuração que apresentou o melhor equilíbrio entre acurácia e eficiência paramétrica foi **ConvLoRA-9 + FullyLoRA-1**. Essa configuração emprega $r = 9$ para o ConvLoRA nas camadas convolucionais e $r = 30$ para o LoRA na camada de classificação, resultando em um total de aproximadamente 1.669.819 parâmetros treináveis. Apesar de configurações como **FullyLoRA-3** atingirem uma acurácia ligeiramente superior (0,76), a diferença em relação à **FullyLoRA-1** (0,75) é marginal, enquanto o número de parâmetros treináveis na **FullyLoRA-1** é significativamente menor. Essa escolha permite um treinamento mais eficiente sem comprometer substancialmente a acurácia do modelo. Assim, a configuração **ConvLoRA-9 + FullyLoRA-1** será adotada como padrão para os experimentos apresentados nas próximas seções desta tese.

5.2.2 Precisão dos Modelos

Para avaliar o desempenho dos adaptadores e do método de referência SenaCNN, utilizamos dois conjuntos de dados amplamente adotados em cenários de aprendizado incremental: CIFAR-100 e CUB-200. Novemante, os experimentos foram conduzidos utilizando a arquitetura VGG-19.

As Tabelas 5.4 e 5.5 apresentam os resultados experimentais, comparando a acurácia final e o número de épocas necessárias para a convergência em cada tarefa. Esses resultados fornecem uma análise detalhada da eficiência dos modelos ao longo do aprendizado incremental, evidenciando a relação entre o número de parâmetros treináveis e a precisão alcançada.

Tabela 5.4. Comparação da acurácia média e número de épocas por tarefa no conjunto de dados CIFAR-100 (variância entre parênteses).

Tarefa	Método	Acurácia Final	Épocas
1	**ConvLoRA**	0,83 (±0,007)	67,00 (±7,329)
	SenaCNN	**0,85 (±0,012)**	**45,13 (±11,934)**
2	**ConvLoRA**	0,88 (±0,008)	41,25 (±5,922)
	SenaCNN	**0,89 (±0,007)**	**31,00 (±4,721)**
3	**ConvLoRA**	0,84 (±0,004)	54,25 (±2,765)
	SenaCNN	**0,84 (±0,008)**	**32,50 (±6,188)**
4	**ConvLoRA**	**0,89 (±0,005)**	48,88 (±6,512)
	SenaCNN	0,89 (±0,005)	**31,88 (±6,978)**
5	**ConvLoRA**	0,85 (±0,005)	53,25 (±4,590)
	SenaCNN	**0,86 (±0,009)**	**37,50 (±5,657)**
6	**ConvLoRA**	0,86 (±0,005)	43,25 (±6,065)
	SenaCNN	**0,86 (±0,006)**	**29,75 (±5,064)**
7	**ConvLoRA**	0,82 (±0,011)	37,88 (±6,289)
	SenaCNN	**0,83 (±0,005)**	**31,13 (±3,482)**
8	**ConvLoRA**	0,84 (±0,005)	50,75 (±6,341)
	SenaCNN	**0,86 (±0,004)**	**36,63 (±2,774)**
9	**ConvLoRA**	0,86 (±0,009)	40,50 (±3,251)
	SenaCNN	**0,86 (±0,007)**	**32,50 (±5,451)**
10	**ConvLoRA**	0,84 (±0,012)	48,75 (±9,677)
	SenaCNN	**0,85 (±0,012)**	**37,75 (±7,536)**

A Figura 5.3 apresenta a evolução da acurácia dos modelos SenaCNN (Figura (a)) e ConvLoRA (Figura (b)) ao longo das 10 tarefas sequenciais do conjunto CIFAR-100. Nota-se que ambos os modelos mantêm níveis estáveis de desempenho mesmo após o aprendizado de novas tarefas, o que indica uma mitigação eficaz do esquecimento catastrófico. Essa ca-

Tabela 5.5. Comparação da acurácia média e número de épocas por tarefa no conjunto de dados CUB-200 (variância entre parênteses).

Tarefa	Método	Acurácia Final	Épocas
1	**ConvLoRA**	0,72 (±0,018)	63,50 (±15,538)
	SenaCNN	**0,76 (±0,004)**	**50,13 (±1,885)**
2	**ConvLoRA**	0,72 (±0,014)	43,50 (±7,309)
	SenaCNN	**0,75 (±0,010)**	**42,13 (±5,540)**
3	**ConvLoRA**	0,70 (±0,010)	41,63 (±4,173)
	SenaCNN	**0,72 (±0,011)**	**38,50 (±5,099)**
4	**ConvLoRA**	**0,79 (±0,026)**	**41,50 (±6,633)**
	SenaCNN	0,79 (±0,017)	43,25 (±5,120)
5	**ConvLoRA**	0,84 (±0,010)	42,38 (±6,632)
	SenaCNN	**0,85 (±0,005)**	**37,50 (±4,899)**

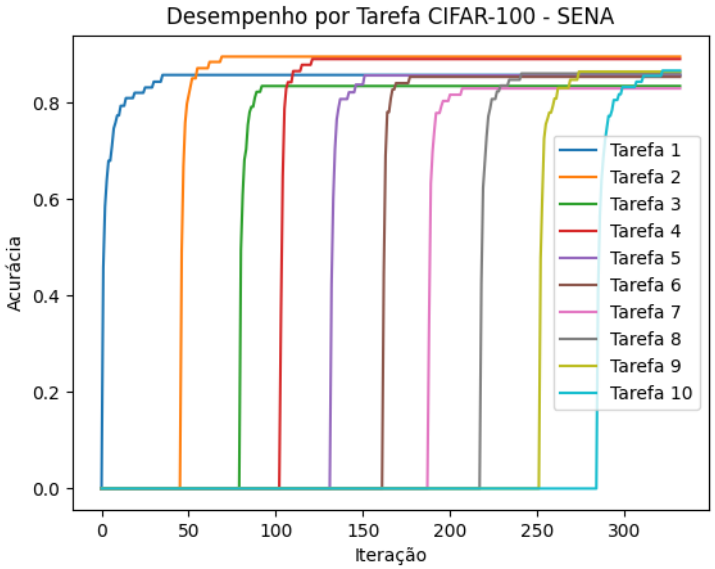
racterística é atribuída à natureza dinâmica dessas abordagens, que protegem os parâmetros relevantes de tarefas anteriores, garantindo uma adaptação contínua sem degradação do conhecimento previamente adquirido.

A Figura 5.4 apresenta a evolução da acurácia dos modelos SenaCNN (Figura (a)) e ConvLoRA (Figura (b)) ao longo das 5 tarefas sequenciais do conjunto CUB-200. Assim como observado no cenário com CIFAR-100, ambos os modelos demonstram estabilidade no desempenho mesmo com a introdução de novas tarefas. Essa resiliência ao esquecimento catastrófico reforça a efetividade das abordagens baseadas em redes dinâmicas, que isolam os parâmetros essenciais de cada tarefa, permitindo aprendizado incremental sem comprometer o conhecimento acumulado.

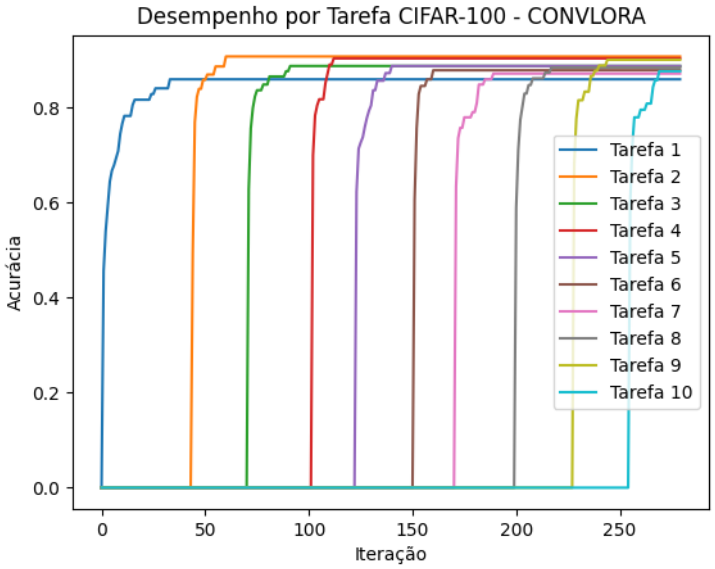
Para o conjunto de dados CIFAR-100 (Tabela 5.4), ambos os métodos apresentaram acurácias finais semelhantes ao longo das tarefas, com o SenaCNN superando ligeiramente o ConvLoRA por uma margem estreita entre 1 e 2%. Essa diferença, embora pequena, é consistente em todas as tarefas, sugerindo que o ajuste fino completo das camadas convolucionais do SenaCNN pode oferecer uma vantagem marginal na capacidade de extração de características. No entanto, essa melhoria ocorre às custas de um número significativamente maior de parâmetros treináveis.

No conjunto de dados CUB-200 (Tabela 5.5), observamos uma tendência semelhante, mas com uma diferença mais pronunciada na primeira tarefa. Essa tarefa inicial representa um desafio maior, pois envolve a aprendizagem de 100 classes simultaneamente, enquanto as quatro tarefas subsequentes contêm apenas 25 classes cada. Esse aumento na carga de aprendizado inicial pode explicar a maior discrepância no desempenho entre os métodos nesse estágio.

Por outro lado, no CIFAR-100, as tarefas foram distribuídas de forma uniforme, com

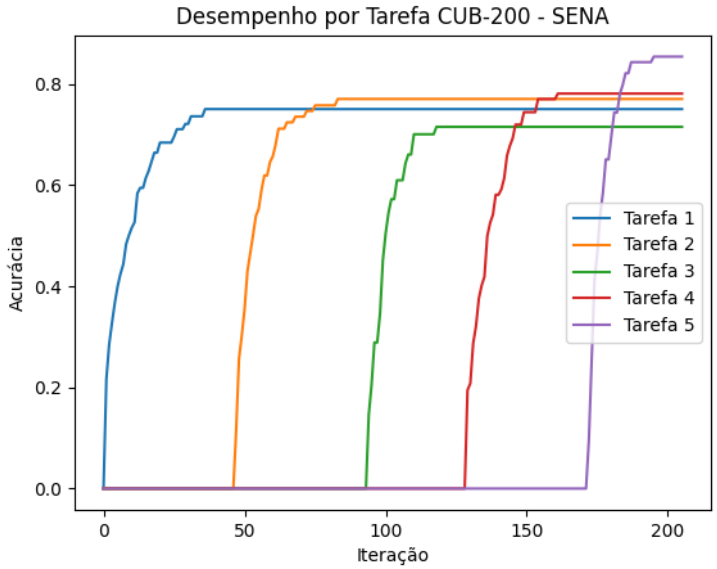


((a)) Desempenho do modelo SenaCNN ao longo das 10 tarefas do CIFAR-100. Observa-se que a acurácia permanece estável após cada tarefa, indicando retenção eficaz do conhecimento.

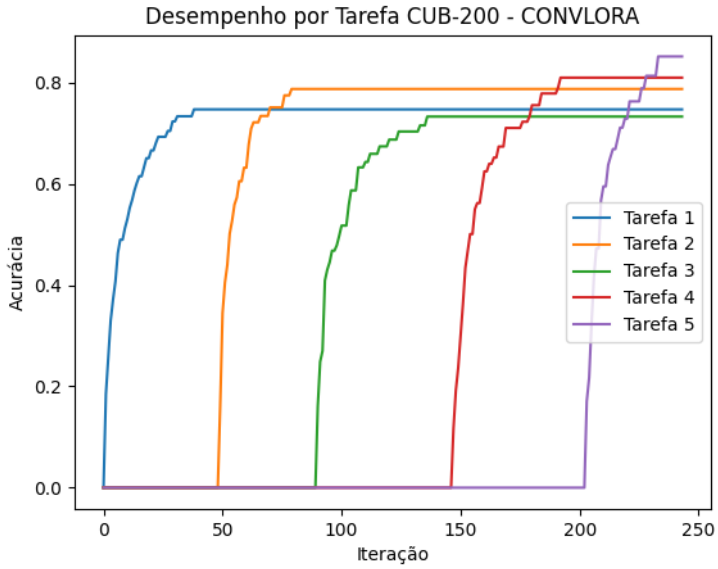


((b)) Desempenho do modelo ConvLoRA no mesmo cenário. Assim como o SenaCNN, o modelo mantém a acurácia ao longo do tempo, sem sinais evidentes de esquecimento catastrófico.

Figura 5.3. Curvas de acurácia ao longo das 10 tarefas do conjunto CIFAR-100 para os modelos SenaCNN ((a)) e ConvLoRA ((b)). Ambos demonstram resiliência ao esquecimento catastrófico devido à sua estrutura baseada em redes dinâmicas, que preservam os parâmetros relevantes de tarefas anteriores.



((a)) Desempenho do modelo SenaCNN ao longo das 5 tarefas do CUB-200. A acurácia permanece elevada após cada nova tarefa, indicando retenção estável do conhecimento aprendido.



((b)) Desempenho do modelo ConvLoRA no mesmo cenário. Assim como o SenaCNN, mantém-se robusto à medida que novas tarefas são aprendidas.

Figura 5.4. Curvas de acurácia ao longo das 5 tarefas do conjunto CUB-200 para os modelos SenaCNN ((a)) e ConvLoRA ((b)). Ambos os modelos demonstram forte resistência ao esquecimento catastrófico, mantendo desempenho estável mesmo com o acréscimo sucessivo de novas tarefas.

cada tarefa contendo 10 classes. Esse balanceamento na divisão das classes ao longo das tarefas parece reduzir a variação no desempenho dos modelos entre as fases de aprendizado. Esses resultados sugerem que a complexidade de tarefas com um maior número de classes pode exigir modelos com maior capacidade de aprendizado, tornando o impacto da estratégia de adaptação ainda mais relevante em cenários onde a distribuição das classes por tarefa é desigual.

O uso de adaptadores, como o ConvLoRA e o LoRA, tem como principal objetivo minimizar o crescimento do número de parâmetros treináveis, mantendo a acurácia próxima ao limite superior estabelecido pelo ajuste fino completo. Como demonstrado nos resultados experimentais, a precisão do modelo usando os adaptadores ficou bem próximo, chegando a superar marginalmente em algumas tarefas.

Além disso, não são necessárias métricas tradicionais de esquecimento para avaliar o desempenho dos modelos testados, uma vez que os adaptadores garantem a preservação do conhecimento adquirido ao longo do treinamento. Diferentemente de abordagens convencionais, onde o aprendizado de novas tarefas pode causar degradação do desempenho em tarefas previamente aprendidas, os adaptadores utilizados asseguram que os parâmetros principais da rede sejam preservados, evitando o fenômeno do esquecimento catastrófico. Dessa forma, a avaliação dos modelos pode ser conduzida exclusivamente com base na acurácia final, no número de épocas necessárias para a convergência e o crescimento do número de parâmetros treinaveis a cada nova tarefa. Essa analise de crescimento será discutida na próxima seção.

5.2.3 Crescimento dos Parâmetros do Modelo

Todos os métodos pertencentes ao grupo de Redes Dinâmicas na literatura que abordam o esquecimento catastrófico (*Catastrophic Forgetting* - CF) apresentam um crescimento no tamanho do modelo a cada nova tarefa aprendida, o que representa uma limitação significativa [Aleixo et al., 2024]. Avaliar o quanto um modelo precisa crescer conforme novas tarefas são incorporadas é essencial, pois esse crescimento implica um aumento na demanda computacional em termos de espaço, tempo e memória.

Idealmente, técnicas que minimizam a adição de novos parâmetros enquanto suportam o aprendizado incremental são preferidas, pois melhoram tanto a eficiência quanto a escalabilidade do modelo. Métodos que conseguem preservar o conhecimento adquirido sem necessidade de expansão excessiva tornam-se particularmente vantajosos em cenários onde os recursos computacionais são limitados.

A Figura 5.5 apresenta a comparação do crescimento cumulativo do modelo considerando todas as camadas. O ConvLoRA demonstra uma eficiência paramétrica significativamente superior ao SenaCNN, necessitando apenas 2,2% dos parâmetros utilizados pelo mé-

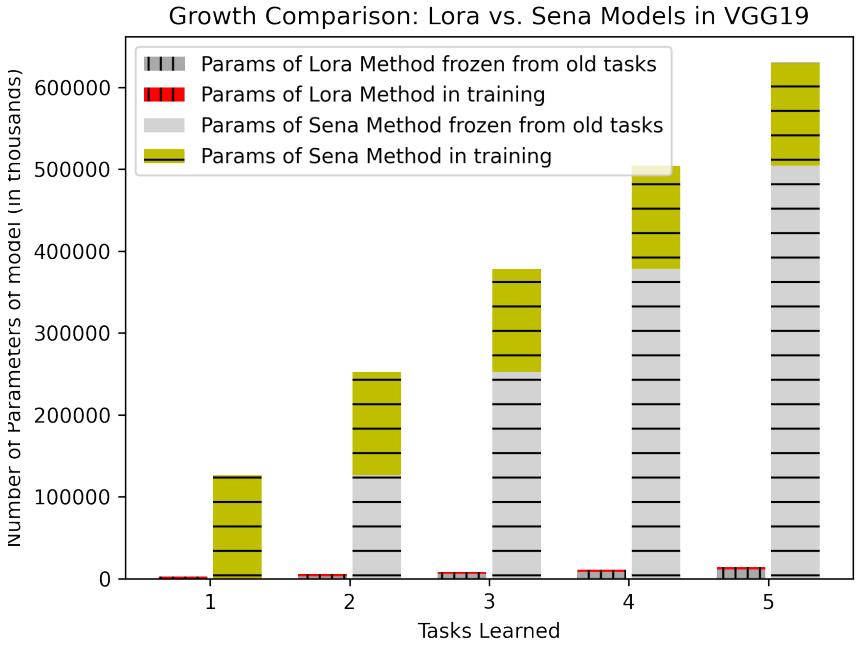


Figura 5.5. Comparação cumulativa do crescimento do tamanho do modelo considerando todas as camadas, quanto menor é o valor melhor é a técnica. As barras com hachura vertical representam o ConvLoRA, enquanto as com hachura horizontal correspondem ao SenaCNN. As seções em cinza indicam a quantidade de parâmetros congelados, enquanto os segmentos em vermelho e verde ilustram os novos parâmetros adicionados a cada nova tarefa. O ConvLoRA demonstra maior eficiência paramétrica, necessitando aproximadamente 2,2% dos parâmetros utilizados pelo SenaCNN.

todo de referência. Já a Figura 5.6 foca especificamente nas camadas convolucionais, onde o ConvLoRA mantém uma economia substancial de parâmetros, utilizando apenas 14,8% dos parâmetros exigidos pelo SenaCNN. Esses resultados evidenciam que, ao longo do aprendizado incremental, o ConvLoRA é capaz de reduzir significativamente a necessidade de novos parâmetros sem comprometer a capacidade do modelo de aprender novas tarefas.

As tendências apresentadas nas Figuras 5.5 e 5.6 destacam a superior relação entre eficiência e crescimento de parâmetros alcançada pelo ConvLoRA em comparação ao SenaCNN. Utilizando a arquitetura VGG-19 como base, o ConvLoRA requer apenas 2,2% do total de parâmetros utilizados pelo SenaCNN por tarefa, com um total de 2.791.099 parâmetros, enquanto o SenaCNN utiliza 126.101.056. Essa redução substancial no crescimento do modelo é evidenciada pela diminuição de 85,17% nos parâmetros das camadas convolucionais (850.419 contra 5.735.800) e uma impressionante redução de 98,39% nos parâmetros da camada classificadora (1.940.680 contra 120.365.256).

É importante ressaltar que as tendências de crescimento são lineares para ambos os métodos, uma vez que adicionam uma quantidade fixa de novos parâmetros para cada tarefa. No entanto, a constante de crescimento do ConvLoRA é significativamente menor. Nossos

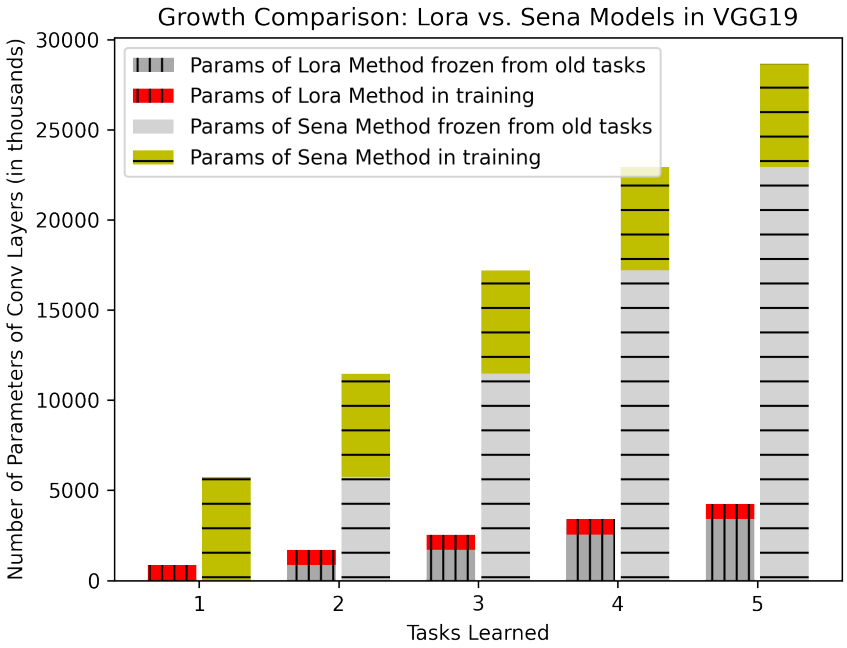


Figura 5.6. Comparação cumulativa do crescimento do tamanho do modelo considerando apenas as camadas convolucionais, quanto menor é o valor melhor é a técnica. O ConvLoRA apresenta economia significativa de parâmetros, utilizando aproximadamente 14,8% dos parâmetros necessários para o SenaCNN.

experimentos indicam que essa redução drástica nos requisitos computacionais não compromete de forma significativa o desempenho da classificação. As tendências ilustradas nas Figuras 5.5 e 5.6 abrangem tanto as camadas convolucionais quanto as camadas de normalização em lote, oferecendo uma avaliação abrangente muitas vezes negligenciada em estudos relacionados.

A eficiência paramétrica superior dos adaptadores torna-se particularmente vantajosa em ambientes com restrições de recursos, demonstrando sua capacidade de escalabilidade e aplicabilidade prática em cenários de aprendizado contínuo. Esses resultados evidenciam que os adaptadores não apenas reduz drasticamente a necessidade de novos parâmetros, mas também garante uma abordagem eficiente e sustentável para o aprendizado incremental.

5.3 Considerações Finais

Neste capítulo, apresentamos uma análise detalhada do impacto dos adaptadores Low-Rank, como ConvLoRA e LoRA, na mitigação do esquecimento catastrófico em redes neurais convolucionais. Através de experimentos conduzidos nos conjuntos de dados CIFAR-100 e CUB-200, investigamos a eficácia dessas técnicas no contexto do aprendizado incremental, comparando-as com o método de referência SenaCNN.

Os resultados demonstraram que o ConvLoRA foi capaz de alcançar uma acurácia comparável ao ajuste fino completo, reduzindo drasticamente o número de parâmetros treináveis. A configuração **ConvLoRA-9 + FullyLoRA-1** emergiu como a mais eficiente, garantindo um desempenho competitivo enquanto minimizava a expansão paramétrica do modelo. Essa abordagem revelou-se particularmente vantajosa em cenários onde a escalabilidade e a eficiência computacional são essenciais.

Além disso, a análise do crescimento dos parâmetros revelou que os adaptadores apresentam uma expansão significativamente mais eficiente em comparação ao SenaCNN. Embora ambos os métodos sigam uma tendência linear na adição de novos parâmetros por tarefa, o ConvLoRA reduz em mais de 85% os parâmetros nas camadas convolucionais e em mais de 98% na camada classificadora. Essa redução minimiza o impacto do aumento de complexidade do modelo ao longo do aprendizado incremental, tornando-o mais adequado para aplicações em ambientes com restrições de memória e processamento.

Outro aspecto relevante dos experimentos foi a constatação de que as métricas tradicionais de esquecimento tornam-se dispensáveis quando utilizamos adaptadores que preservam a informação das tarefas anteriores sem necessidade de reconfiguração dos parâmetros da rede principal. Isso simplifica a análise do aprendizado incremental e reforça a viabilidade do uso de adaptadores para redes convolucionais.

Com base nesses achados, este estudo contribui para a discussão sobre métodos eficientes de aprendizado incremental, apontando que abordagens baseadas em adaptadores são uma solução promissora para mitigar o esquecimento catastrófico sem comprometer o desempenho do modelo. No próximo capítulo, apresentamos a conclusão da tese, onde discutiremos as implicações desses resultados, bem como possíveis direções para trabalhos futuros.

6

Conclusões

Nesta tese, abordamos o problema do esquecimento catastrófico em redes neurais convolucionais (RNC) no contexto de aprendizado contínuo, explorando a eficácia das abordagens Low-Rank Adapters (LoRA) e ConvLoRA. O objetivo principal da pesquisa foi verificar como uma RNC pode aprender novas tarefas sequencialmente sem perder a qualidade nas tarefas previamente aprendidas, mantendo a eficiência na adição de novos parâmetros e eliminando a necessidade de armazenar dados de tarefas anteriores. Para isso, analisamos três perguntas de pesquisa centrais formuladas na introdução:

1. **É possível um modelo aprender novas tarefas, sequencialmente, sem perder qualidade nas tarefas já aprendidas?**

   Sim, a implementação do ConvLoRA provou que é possível aprender novas tarefas enquanto preserva a precisão em tarefas anteriores. Ao aplicar técnicas de adaptação de baixa ordem, garantimos que, mesmo com a incorporação de novas classes, o modelo manteve alta performance nas classes previamente treinadas.

2. **Existe um limite no número de tarefas que podem ser aprendidas?**

   As experiências realizadas sugerem que, embora o número de tarefas que um modelo possa aprender seja geralmente limitado pela capacidade de aprendizado deste, a abordagem proposta minimiza o tamanho do modelo, permitindo ao mesmo suportar um número maior de tarefas de maneira eficiente. Por meio da introdução de apenas um baixo aumento no número de parâmetros a cada nova tarefa, conseguimos aumentar a escalabilidade e a adaptabilidade do modelo.

3. **A ordem em que as tarefas são aprendidas interfere na retenção do conhecimento?**

Embora esta pesquisa não tenha se aprofundado especificamente na análise da ordem das tarefas, as observações feitas durante os experimentos indicaram que a ordem das tarefas pode ter um impacto no desempenho, especialmente em cenários com tarefas altamente correlacionadas. Contudo, a preservação do conhecimento das tarefas anteriores é garantida pela natureza da metodologia implementada, que se baseia aplicar adaptadores específicos para cada tarefas.

Através dos métodos de ConvLoRA e LoRA, foi demonstrado que é possível mitigar os efeitos do esquecimento catastrófico, permitindo que modelos sejam treinados de maneira contínua. Os resultados obtidos com os conjuntos de dados CIFAR-100 e CUB-200 demonstraram que, embora os adaptadores tenham fornecido uma acurácia comparável em diversas tarefas em relação ao método de referência SenaCNN, eles exigiram uma quantidade significativamente menor de parâmetros. Mais especificamente, observou-se uma redução superior a 98% no número total de parâmetros para cada nova tarefa, refletindo a eficiência do modelo em termos de memória e tempo de treinamento.

6.1 Publicações

Durante o desenvolvimento deste trabalho, foram produzidos diversos artefatos científicos que contribuíram para a fundamentação teórica e validação experimental da pesquisa, dentre os quais podemos destacar os dois principais a seguir.

O primeiro resultado foi um estudo abrangente sobre esquecimento catastrófico em redes neurais profundas, publicado no artigo *Catastrophic Forgetting in Deep Learning: A Comprehensive Taxonomy* [Aleixo et al., 2024], na edição de 2024 do *Journal of the Brazilian Computer Society*. Esse estudo resultou na taxonomia apresentada no Capítulo 3, que categoriza as abordagens existentes para mitigar o esquecimento catastrófico. Essa estrutura conceitual foi essencial para guiar as definições e decisões metodológicas no desenvolvimento do modelo proposto no Capítulo 4, alinhando-o à categoria de redes dinâmicas.

Além dessa contribuição teórica, desenvolvemos e avaliamos um novo método para aprendizado incremental baseado em redes dinâmicas. Esse método se destaca por equilibrar eficiência computacional e desempenho na preservação do conhecimento adquirido ao longo de múltiplas tarefas. Os resultados experimentais demonstraram vantagens significativas em comparação com abordagens convencionais. O artigo descrevendo esse método encontra-se em processo de submissão à revista *PeerJ Computer Science*, reforçando a relevância da proposta no contexto do aprendizado contínuo.

Dessa forma, os artefatos produzidos não apenas consolidam a fundamentação teórica deste trabalho, mas também ampliam suas aplicações, contribuindo para a evolução das

técnicas de aprendizado contínuo em diferentes domínios.

Por fim, durante o programa de doutorado também foram realizadas outras publicações não relacionadas diretamente com o objeto desta tese, seja em função de disciplinas realizadas no programa ou do trabalho realizado no Sidia Instituto de Pesquisa e Desenvolvimento. São esses:

- **SVC-A2C-Actor Critic Algorithm to Improve Smart Vacuum Cleaner** *Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC)*, pp. 65-70, 2019[Aleixo et al., 2019].
- **Using String-Comparison Measures to Improve and Evaluate Collaborative Filtering Recommender Systems** *Bias and Social Aspects in Search and Recommendation: First International Conference*, 2020[Pascoal et al., 2020].
- **Visual prediction based on photorealistic style transfer** *International Conference on Human-Computer Interaction*, pp. 301-309, 2021 [Fernandes et al., 2021].
- **Designing 3D Avatar Influencer for Live Streaming Interactions** *International Conference on Human-Computer Interaction*, pp. 41-51, 2024[Lourenço et al., 2024].
- **Modular Platform for Health and Safety Data Monitoring** *International Conference on Human-Computer Interaction*, pp. 30-39, 2025 [Marques et al., 2025].

6.2 Trabalhos Futuros

Com base nos achados desta pesquisa, algumas direções para trabalhos futuros podem ser propostas:

- **Exploração do impacto da ordem das tarefas:** A realização de experimentos sistemáticos para avaliar como a sequência em que as tarefas são aprendidas pode influenciar a retenção do conhecimento serve como foco. Tal análise pode revelar *insights* adicionais sobre a dinâmica de aprendizado de máquinas e otimização de modelos.
- **Aprimoramento dos adaptadores:** Investigar novas arquiteturas de adaptadores que aumentem ainda mais a eficiência do treinamento, considerando não apenas as dimensões de low-rank, mas também técnicas que abordem a modularidade e a especialização dos módulos adaptativos.
- **Geração de adaptadores pós-treino:** Conforme demonstrado no capítulo de resultados, a estratégia de ajuste fino em todo o modelo alcança uma precisão superior

em comparação aos adaptadores treinados separadamente. Dessa forma, propomos investigar a viabilidade de um método para a extração de adaptadores a partir de modelos previamente ajustados. Especificamente, pretende-se explorar a possibilidade de treinar o modelo em uma nova tarefa, calcular a diferença (*delta*) entre os pesos do modelo ajustado e sua versão inicial e, com base nessa variação, empregar técnicas de fatoração para identificar representações compactas que preservem a qualidade da inferência. O objetivo dessa abordagem é obter adaptadores que mantenham um desempenho comparável ao ajuste fino integral do modelo, minimizando, entretanto, a introdução de novos parâmetros para cada tarefa aprendida.

- **Aplicações em cenários reais:** Avaliar a robustez do ConvLoRA em sistemas de aprendizado contínuo aplicados a cenários reais. Como estudo de caso, implementaremos o modelo para o reconhecimento de aves em diferentes regiões do mundo, analisando sua capacidade de adaptação a novos domínios geográficos. O objetivo é validar a eficácia do método diante de variações ambientais e diferenças na distribuição das classes, contribuindo para aplicações práticas em biodiversidade e conservação.

- **Integração de mecanismos de atenção:** Considerar a incorporação de mecanismos de atenção nas redes para melhorar a captura de características importantes em diferentes classes, potencializando ainda mais a capacidade de aprendizagem das RNC em ambientes dinâmicos e com múltiplas tarefas.

Em conclusão, esta pesquisa contribui para o avanço do conhecimento na área de aprendizado contínuo em RNCs, apresentando soluções que podem revolucionar a forma como sistemas de inteligência artificial são projetados para operar em contextos adaptativos. Em um mundo cada vez mais dinâmico, a capacidade de aprender continuamente sem perda de conhecimento anterior torna-se não apenas uma vantagem, mas uma necessidade para a evolução da inteligência artificial.

Referências Bibliográficas

Adel, T.; Zhao, H. & Turner, R. E. (2020). Continual learning with adaptive weights (claw). Em *International Conference on Learning Representations*.

Aggarwal, A.; Mittal, M. & Battineni, G. (2021). Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, 1(1):100004.

Ahn, H.; Kwak, J.; Lim, S.; Bang, H.; Kim, H. & Moon, T. (2021). Ss-il: Separated softmax for incremental learning. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 844--853.

Aleem, S.; Dietlmeier, J.; Arazo, E. & Little, S. (2024). Convlora and adabn based domain adaptation via self-training. *2024 IEEE International Symposium on Biomedical Imaging (ISBI)*, pp. 1–5.

Aleixo, E.; Colonna, J. & Barreto, R. (2019). Svc-a2c - actor critic algorithm to improve smart vacuum cleaner. Em *Anais Estendidos do Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC)*, pp. 65--70.

Aleixo, E. L.; Colonna, J. G.; Cristo, M. & Fernandes, E. (2024). Catastrophic forgetting in deep learning: A comprehensive taxonomy. *Journal of the Brazilian Computer Society*, 30(1):175–211.

Aljundi, R.; Chakravarty, P. & Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3366--3375.

Aljundi, R.; Kelchtermans, K. & Tuytelaars, T. (2019). Task-free continual learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11254--11263.

Aloysius, N. & Geetha, M. (2017). A review on deep convolutional neural networks. Em *2017 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0588–0592.

Ashfahani, A. & Pratama, M. (2019). Autonomous deep learning: Continual learning approach for dynamic environments. Em Berger-Wolf, T. Y. & Chawla, N. V., editores, *Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019, Calgary, Alberta, Canada, May 2-4, 2019*, pp. 666--674. SIAM.

Banayeeanzade, M.; Mirzaiezadeh, R.; Hasani, H. & Soleymani, M. (2021). Generative vs. discriminative: Rethinking the meta-continual learning. *Advances in Neural Information Processing Systems*, 34.

Belouadah, E. & Popescu, A. (2019). Il2m: Class incremental learning with dual memory. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 583--592.

Belouadah, E. & Popescu, A. (2020). Scail: Classifier weights scaling for class incremental learning. Em *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1266--1275.

Belouadah, E.; Popescu, A. & Kanellos, I. (2020). Initial classifier weights replay for memoryless class incremental learning. Em *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blauch, N.; Behrmann, M. & Plaut, D. (2020). Computational insights into human perceptual expertise for familiar and unfamiliar face recognition. *Cognition*, 208:104341.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K. & Wierstra, D. (2015). Weight uncertainty in neural network. Em Bach, F. R. & Blei, D. M., editores, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1613--1622. JMLR.org.

Borsos, Z.; Mutny, M. & Krause, A. (2020). Coresets via bilevel optimization for continual learning and streaming. Em Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. & Lin, H., editores, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford,

A.; Sutskever, I. & Amodei, D. (2020). Language models are few-shot learners. *CoRR*, abs/2005.14165.

Buzzega, P.; Boschini, M.; Porrello, A.; Abati, D. & CALDERARA, S. (2020). Dark experience for general continual learning: a strong, simple baseline. Em Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F. & Lin, H., editores, *Advances in Neural Information Processing Systems*, volume 33, pp. 15920--15930. Curran Associates, Inc.

Caccia, M.; Rodríguez, P.; Ostapenko, O.; Normandin, F.; Lin, M.; Page-Caccia, L.; Laradji, I. H.; Rish, I.; Lacoste, A.; Vázquez, D. & Charlin, L. (2020). Online fast adaptation and knowledge accumulation (OSAKA): a new approach to continual learning. Em Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. & Lin, H., editores, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Cai, S.; Xu, Z.; Huang, Z.; Chen, Y. & Kuo, C. J. (2018). Enhancing CNN incremental learning capability with an expanded network. Em *2018 IEEE International Conference on Multimedia and Expo, ICME 2018, San Diego, CA, USA, July 23-27, 2018*, pp. 1--6. IEEE Computer Society.

Cai, Z.; Sener, O. & Koltun, V. (2021). Online continual learning with natural distribution shifts: An empirical study with visual data. *arXiv preprint arXiv:2108.09020*.

Candès, E. J. & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717--772.

Castro, F. M.; Marín-Jiménez, M. J.; Guil, N.; Schmid, C. & Alahari, K. (2018). End-to-end incremental learning. Em Ferrari, V.; Hebert, M.; Sminchisescu, C. & Weiss, Y., editores, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, volume 11216 of *Lecture Notes in Computer Science*, pp. 241--257. Springer.

Cermelli, F.; Mancini, M.; Bulo, S. R.; Ricci, E. & Caputo, B. (2020). Modeling the background for incremental learning in semantic segmentation. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9233--9242.

Cha, H.; Lee, J. & Shin, J. (2021). Co2l: Contrastive continual learning. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9516–9525.

Chaudhry, A.; Ranzato, M.; Rohrbach, M. & Elhoseiny, M. (2019a). Efficient lifelong learning with a-GEM. Em *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Chaudhry, A.; Rohrbach, M.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H. S. & Ranzato, M. (2019b). Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486.

Chen, K. & Lee, C.-G. (2021). Incremental few-shot learning via vector quantization in deep embedded space. Em *International Conference on Learning Representations*.

Chen, X. & He, K. (2021). Exploring simple siamese representation learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750--15758.

Cheraghian, A.; Rahman, S.; Fang, P.; Roy, S. K.; Petersson, L. & Harandi, M. (2021). Semantic-aware knowledge distillation for few-shot class-incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2534--2543.

Church, K. W. (2017). Word2vec. *Natural Language Engineering*, 23(1):155–162.

Clark, C. & Storkey, A. (2015). Training deep convolutional neural networks to play go. Em *International conference on machine learning*, pp. 1766--1774.

Clune, J.; Mouret, J. & Lipson, H. (2013). Summary of the evolutionary origins of modularity. Em Blum, C. & Alba, E., editores, *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, Companion Material Proceedings*, pp. 23--24. ACM.

Coop, R.; Mishtal, A. & Arel, I. (2013). Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting. *IEEE Trans. Neural Networks Learn. Syst.*, 24(10):1623--1634.

Davari, M.; Asadi, N.; Mudur, S.; Aljundi, R. & Belilovsky, E. (2022). Probing representation forgetting in supervised and unsupervised continual learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022)*.

De Lange, M. & Tuytelaars, T. (2021). Continual prototype evolution: Learning online from non-stationary data streams. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8250--8259.

Devlin, J.; Chang, M.; Lee, K. & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. Em Burstein, J.; Doran, C. & Solorio, T., editores, *Proceedings of the 2019 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pp. 4171--4186. Association for Computational Linguistics.

Dhar, P.; Singh, R. V.; Peng, K.-C.; Wu, Z. & Chellappa, R. (2019). Learning without memorizing. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Diethe, T.; Borchert, T.; Thereska, E.; Balle, B. & Lawrence, N. (2019). Continual learning in practice. *arXiv preprint arXiv:1903.05202*.

Dong, N.; Zhang, Y.; Ding, M. & Lee, G. H. (2021). Bridging non co-occurrence with unlabeled in-the-wild data for incremental object detection. *Advances in Neural Information Processing Systems*, 34.

Douillard, A.; Ramé, A.; Couairon, G. & Cord, M. (2022). Dytox: Transformers for continual learning with dynamic token expansion. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9285–9295.

Egorov, E.; Kuzina, A. & Burnaev, E. (2021). Boovae: Boosting approach for continual learning of vae. *Advances in Neural Information Processing Systems*, 34.

Ellefsen, K. O.; Mouret, J. & Clune, J. (2015). Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Comput. Biol.*, 11(4).

Fernandes, E.; Aleixo, E.; Barreira, W. J.; Gadelha, M. R.; Khurshid, A. & Tamayo, S. C. (2021). Visual prediction based on photorealistic style transfer. Em Degen, H. & Ntoa, S., editores, *Artificial Intelligence in HCI*, pp. 301--309, Cham. Springer International Publishing.

Fernando, C.; Banarse, D.; Blundell, C.; Zwols, Y.; Ha, D.; Rusu, A. A.; Pritzel, A. & Wierstra, D. (2017). Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Gao, Z.; Xu, C.; Li, F.; Jia, Y.; Harandi, M. & Wu, Y. (2023). Exploring data geometry for continual learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24325–24334.

Gepperth, A. & Gondal, S. A. (2018). Incremental learning with deep neural networks using a test-time oracle. Em *26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium, April 25-27, 2018*.

Gepperth, A.; Lefort, M. & Hecht, T. (2015). Resource-efficient incremental learning in very high dimensions. Em *23rd European Symposium on Artificial Neural Networks, ESANN 2015, Bruges, Belgium, April 22-24, 2015*, Bruges, Belgium.

Golub, G. H. & Van Loan, C. F. (1996). *Matrix Computations*. Johns Hopkins University Press.

Goodfellow, I. J.; Bengio, Y. & Courville, A. C. (2016). *Deep Learning*. Adaptive computation and machine learning. MIT Press.

Goodfellow, I. J.; Mirza, M.; Da, X.; Courville, A. C. & Bengio, Y. (2014a). An empirical investigation of catastrophic forgeting in gradient-based neural networks. Em Bengio, Y. & LeCun, Y., editores, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C. & Bengio, Y. (2014b). Generative adversarial nets. Em Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D. & Weinberger, K. Q., editores, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2672--2680.

Goodrich, B. & Arel, I. (2014). Neuron clustering for mitigating catastrophic forgetting in feedforward neural networks. Em *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, CIDUE 2014, Orlando, FL, USA, December 9-12, 2014*, pp. 62--68. IEEE.

Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D. J. & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. Em Bach, F. R. & Blei, D. M., editores, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1462--1471, Lille, France. PMLR.

Greve, R. B.; Jacobsen, E. J. & Risi, S. (2016). Evolving neural turing machines for reward-based learning. Em Friedrich, T.; Neumann, F. & Sutton, A. M., editores, *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, p. 117–124, New York, NY, USA. Association for Computing Machinery.

Guo, H.; Yan, Z.; Yang, J. & Li, S. (2018). An incremental scheme with weight pruning to train deep neural network. Em Liang, Q.; Liu, X.; Na, Z.; Wang, W.; Mu, J. & Zhang, B., editores, *Communications, Signal Processing, and Systems - Proceedings of the 2018 CSPS, Volume III: Systems, Dalian, China, 14-16 July 2018*, volume 517 of *Lecture Notes in Electrical Engineering*, pp. 295--302. Springer.

Han, S.; Pool, J.; Narang, S.; Mao, H.; Gong, E.; Tang, S.; Elsen, E.; Vajda, P.; Paluri, M.; Tran, J.; Catanzaro, B. & Dally, W. J. (2017). DSD: dense-sparse-dense training for deep neural networks. Em *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Han, S.; Pool, J.; Tran, J. & Dally, W. J. (2015). Learning both weights and connections for efficient neural network. Em Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M. & Garnett, R., editores, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1135--1143.

Hattori, M. & Tsuboi, H. (2018). Reduction of catastrophic forgetting for multilayer neural networks trained by no-prop algorithm. Em *2018 International Conference on Information and Communications Technology (ICOIACT)*, pp. 214–219. IEEE.

Hayes, T. L.; Cahill, N. D. & Kanan, C. (2019). Memory efficient experience replay for streaming learning. Em *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pp. 9769--9776. IEEE.

Hayes, T. L.; Kafle, K.; Shrestha, R.; Acharya, M. & Kanan, C. (2020). REMIND your neural network to prevent catastrophic forgetting. Em Vedaldi, A.; Bischof, H.; Brox, T. & Frahm, J., editores, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VIII*, volume 12353 of *Lecture Notes in Computer Science*, pp. 466--483. Springer.

He, C.; Wang, R.; Shan, S. & Chen, X. (2018). Exemplar-supported generative reproduction for class incremental learning. Em *British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*, p. 98. BMVA Press.

He, K.; Zhang, X.; Ren, S. & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Em *Proceedings of the IEEE international conference on computer vision*, pp. 1026--1034.

He, K.; Zhang, X.; Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. Em *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770--778. IEEE Computer Society.

Henning, C.; Cervera, M.; D'Angelo, F.; Von Oswald, J.; Traber, R.; Ehret, B.; Kobayashi, S.; Grewe, B. F. & Sacramento, J. (2021). Posterior meta-replay for continual learning. *Advances in Neural Information Processing Systems*, 34.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I. & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580. cite arxiv:1207.0580.

Hinton, G. E.; Vinyals, O. & Dean, J. (2015). Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J. et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hong, D.; Li, Y. & Shin, B.-S. (2019). Predictive ewc: mitigating catastrophic forgetting of neural network through pre-prediction of learning data. *Journal of Ambient Intelligence and Humanized Computing*, pp. 1--10.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z. & Lin, D. (2018). Lifelong learning via progressive distillation and retrospection. Em *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 437--452.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z. & Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hu, E. J.; yelong shen; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L. & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. Em *International Conference on Learning Representations*.

Hu, W.; Lin, Z.; Liu, B.; Tao, C.; Tao, Z.; Ma, J.; Zhao, D. & Yan, R. (2019). Overcoming catastrophic forgetting for continual learning via model adaptation. Em *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Hu, X.; Tang, K.; Miao, C.; Hua, X.-S. & Zhang, H. (2021). Distilling causal effect of data in class-incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3957--3966.

Hu, Z.; Li, Y.; Lyu, J.; Gao, D. & Vasconcelos, N. (2023). Dense network expansion for class incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11858–11867.

Huang, G.; Zhu, Q. & Siew, C. K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489--501.

Huang, Y.; Zhang, Y.; Chen, J.; Wang, X. & Yang, D. (2021). Continual learning for text classification with information disentanglement based regularization. Em *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2736--2746, Online. Association for Computational Linguistics.

Hung, C.-Y.; Tu, C.-H.; Wu, C.-E.; Chen, C.-H.; Chan, Y.-M. & Chen, C.-S. (2019). Compacting, picking and growing for unforgetting continual learning. Em Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. & Garnett, R., editores, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Huynh, B. Q.; Li, H. & Giger, M. L. (2016). Digital mammographic tumor classification using transfer learning from deep convolutional neural networks. *Journal of Medical Imaging*, 3(3):034501.

Imai, S. & Nobuhara, H. (2018). Stepwise pathnet: Transfer learning algorithm to improve network structure versatility. Em *IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018, Miyazaki, Japan, October 7-10, 2018*, pp. 918--922. IEEE.

Javed, K. & White, M. (2019). Meta-learning representations for continual learning. Em Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. & Garnett, R., editores, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jin, X.; Sadhu, A.; Du, J. & Ren, X. (2021). Gradient-based editing of memory examples for online task-free continual learning. *Advances in Neural Information Processing Systems*, 34.

Joseph, K. J. & Balasubramanian, V. N. (2020). Meta-consolidation for continual learning. Em Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. & Lin, H., editores, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jung, S.; Ahn, H.; Cha, S. & Moon, T. (2020). Continual learning with node-importance based adaptive group sparse regularization. Em Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. & Lin, H., editores, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kaplanis, C.; Shanahan, M. & Clopath, C. (2019). Policy consolidation for continual reinforcement learning. Em Chaudhuri, K. & Salakhutdinov, R., editores, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3242--3251. PMLR.

Karatzas, I. & Shreve, S. E. (1991). *Brownian Motion and Stochastic Calculus*. Graduate Texts in Mathematics (113) (Book 113). Springer New York.

Ke, Z.; Liu, B. & Huang, X. (2020). Continual learning of a mixed sequence of similar and dissimilar tasks. Em Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. & Lin, H., editores, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ke, Z.; Xu, H. & Liu, B. (2021). Adapting BERT for continual learning of a sequence of aspect sentiment classification tasks. Em *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4746--4755, Online. Association for Computational Linguistics.

Kemker, R.; McClure, M.; Abitino, A.; Hayes, T. L. & Kanan, C. (2018). Measuring catastrophic forgetting in neural networks. Em McIlraith, S. A. & Weinberger, K. Q., editores, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3390--3398. AAAI Press.

Kim, G.; Liu, B. & Ke, Z. (2022). A multi-head model for continual learning via out-of-distribution replay. Em Chandar, S.; Pascanu, R. & Precup, D., editores, *Conference on Lifelong Learning Agents, CoLLAs 2022, 22-24 August 2022, McGill University, Montréal, Québec, Canada*, volume 199 of *Proceedings of Machine Learning Research*, pp. 548--563. PMLR.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N. C.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Ku-

maran, D. & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521--3526.

Knoblauch, J.; Husain, H. & Diethe, T. (2020). Optimal continual learning has perfect memory and is NP-hard. Em III, H. D. & Singh, A., editores, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5327--5337. PMLR.

Kobayashi, T. (2018). Check regularization: Combining modularity and elasticity for memory consolidation. Em Kurková, V.; Manolopoulos, Y.; Hammer, B.; Iliadis, L. S. & Maglogiannis, I., editores, *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part II*, volume 11140 of *Lecture Notes in Computer Science*, pp. 315--325. Springer.

Koren, Y.; Bell, R. & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30--37.

Krizhevsky, A.; Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Em Bartlett, P. L.; Pereira, F. C. N.; Burges, C. J. C.; Bottou, L. & Weinberger, K. Q., editores, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 1106--1114.

Lancewicki, T.; Goodrich, B. & Arel, I. (2015). Sequential covariance-matrix estimation with application to mitigating catastrophic forgetting. Em Li, T.; Kurgan, L. A.; Palade, V.; Goebel, R.; Holzinger, A.; Verspoor, K. & Wani, M. A., editores, *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*, pp. 628--633. IEEE.

LeCun, Y.; Bengio, Y. & Hinton, G. E. (2015). Deep learning. *Nat.*, 521(7553):436--444.

LeCun, Y.; Bottou, L.; Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Ledoit, O.; Wolf, M. et al. (2012). Nonlinear shrinkage estimation of large-dimensional covariance matrices. *The Annals of Statistics*, 40(2):1024 -- 1060.

Lee, K.; Lee, K.; Shin, J. & Lee, H. (2019). Overcoming catastrophic forgetting with unlabeled data in the wild. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 312--321.

Lee, S.; Ha, J.; Zhang, D. & Kim, G. (2020). A neural dirichlet process mixture model for task-free continual learning. Em *International Conference on Learning Representations*.

Lee, S.; Kim, J.; Jun, J.; Ha, J. & Zhang, B. (2017a). Overcoming catastrophic forgetting by incremental moment matching. Em Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N. & Garnett, R., editores, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4652--4662.

Lee, S.; Lee, C.; Kwak, D.; Ha, J.; Kim, J. & Zhang, B. (2017b). Dual-memory neural networks for modeling cognitive activities of humans via wearable sensors. *Neural Networks*, 92:17--28.

Leontev, M. I.; Mikheev, A.; Sviatov, K. & Sukhov, S. V. (2019). Overcoming catastrophic interference with bayesian learning and stochastic langevin dynamics. Em Lu, H.; Tang, H. & Wang, Z., editores, *Advances in Neural Networks - ISNN 2019 - 16th International Symposium on Neural Networks, ISNN 2019, Moscow, Russia, July 10-12, 2019, Proceedings, Part I*, volume 11554 of *Lecture Notes in Computer Science*, pp. 370--378. Springer.

Li, T.; Ke, Q.; Rahmani, H.; Ho, R. E.; Ding, H. & Liu, J. (2021). Else-net: Elastic semantic network for continual action recognition from skeleton data. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13434--13443.

Li, X.; Zhou, Y.; Wu, T.; Socher, R. & Xiong, C. (2019). Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. Em Chaudhuri, K. & Salakhutdinov, R., editores, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3925--3934. PMLR.

Li, Z. & Hoiem, D. (2018). Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947.

Liang, D.; Yang, F.; Zhang, T. & Yang, P. (2018). Understanding mixup training methods. *IEEE Access*, 6:58774--58783.

Lipson, H. (2007). Principles of modularity, regularity, and hierarchy for scalable systems. *Journal of Biological Physics and Chemistry*, 7(4):125.

Liu, H.; Gu, L.; Chi, Z.; Wang, Y.; Yu, Y.; Chen, J. & Tang, J. (2022). Few-shot class-incremental learning via entropy-regularized data-free replay. Em Avidan, S.; Brostow,

G.; Cissé, M.; Farinella, G. M. & Hassner, T., editores, *Computer Vision – ECCV 2022*, pp. 146--162, Cham. Springer Nature Switzerland.

Liu, Q.; Majumder, O.; Achille, A.; Ravichandran, A.; Bhotika, R. & Soatto, S. (2020a). Incremental few-shot meta-learning via indirect discriminant alignment. Em Vedaldi, A.; Bischof, H.; Brox, T. & Frahm, J.-M., editores, *Computer Vision – ECCV 2020*, pp. 685--701, Cham. Springer International Publishing.

Liu, X.; Masana, M.; Herranz, L.; Van de Weijer, J.; López, A. M. & Bagdanov, A. D. (2018). Rotate your networks: Better weight consolidation and less catastrophic forgetting. Em *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2262–2268.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L. & Stoyanov, V. (2020b). Ro{bert}a: A robustly optimized {bert} pretraining approach.

Liu, Y.; Schiele, B. & Sun, Q. (2021). Rmm: Reinforced memory management for class-incremental learning. *Advances in Neural Information Processing Systems*, 34.

Liu, Y.; Su, Y.; Liu, A.-A.; Schiele, B. & Sun, Q. (2020c). Mnemonics training: Multi-class incremental learning without forgetting. Em *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 12245--12254.

Lopez-Paz, D. & Ranzato, M. A. (2017). Gradient episodic memory for continual learning. Em Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S. & Garnett, R., editores, *Advances in Neural Information Processing Systems*, volume 30, pp. 6467--6476. Curran Associates, Inc.

Lourenço, A.; Aleixo, E.; Nogueira, M.; Moraes, R.; Dutra, B.; Penalber, G. & Teófilo, M. (2024). Designing 3d avatar influencer for live streaming interactions. Em Stephanidis, C.; Antona, M.; Ntoa, S. & Salvendy, G., editores, *HCI International 2024 Posters*, pp. 41--51, Cham. Springer Nature Switzerland.

Lu, L.; Zheng, Y.; Carneiro, G. & Yang, L. (2017). Deep learning and convolutional neural networks for medical image computing. *Advances in Computer Vision and Pattern Recognition; Springer: New York, NY, USA*.

Lüders, B.; Schläger, M.; Korach, A. & Risi, S. (2017). Continual and one-shot learning through neural networks with dynamic external memory. Em Squillero, G. & Sim, K., editores, *Applications of Evolutionary Computation - 20th European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Part I*, volume 10199 of *Lecture Notes in Computer Science*, pp. 886--901.

Ma, C.; Ji, Z.; Huang, Z.; Shen, Y.; Gao, M. & Xu, J. (2023). Progressive voronoi diagram subdivision enables accurate data-free class-incremental learning. Em *The Eleventh International Conference on Learning Representations*.

Madaan, D.; Yoon, J.; Li, Y.; Liu, Y. & Hwang, S. J. (2022). Representational continuity for unsupervised continual learning. Em *10th International Conference on Learning Representations, ICLR 2022, Virtual, April 25 - April 29, 2022, Conference Track Proceedings*.

Mai, Z.; Li, R.; Kim, H. & Sanner, S. (2021). Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3589--3599.

Mallya, A.; Davis, D. & Lazebnik, S. (2018). Piggyback: Adapting a single network to multiple tasks by learning to mask weights. Em Ferrari, V.; Hebert, M.; Sminchisescu, C. & Weiss, Y., editores, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, volume 11208 of *Lecture Notes in Computer Science*, pp. 72--88. Springer.

Mallya, A. & Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. Em *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 7765--7773. IEEE Computer Society.

Marques, J.; Teofilo, M.; Aleixo, E.; Filho, F.; Díaz, A. A. O. & Cleger Tamayo, S. (2025). Modular platform for health and safety data monitoring. Em Stephanidis, C.; Antona, M.; Ntoa, S. & Salvendy, G., editores, *HCI International 2024 – Late Breaking Posters*, pp. 30--39, Cham. Springer Nature Switzerland.

Masana, M.; Liu, X.; Twardowski, B.; Menta, M.; Bagdanov, A. D. & van de Weijer, J. (2020). Class-incremental learning: survey and performance evaluation. *CoRR*, abs/2010.15277.

Masse, N. Y.; Grant, G. D. & Freedman, D. J. (2018). Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proc. Natl. Acad. Sci. USA*, 115(44):E10467--E10475.

McCloskey, M. & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. Em Bower, G. H., editor, *Psychology of Learning and Motivation*, volume 24, pp. 109--165. Academic Press.

Mellado, D.; Saavedra, C.; Chabert, S. & Salas, R. (2017). Pseudorehearsal approach for incremental learning of deep convolutional neural networks. Em Barone, D. A. C.; Teles, E. O. & Brackmann, C. P., editores, *Computational Neuroscience*, pp. 118--126, Cham. Springer International Publishing.

Mermillod, M.; Bugaiska, A. & Bonin, P. (2013). The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504.

Mi, F.; Chen, L.; Zhao, M.; Huang, M. & Faltings, B. (2020a). Continual learning for natural language generation in task-oriented dialog systems. Em Cohn, T.; He, Y. & Liu, Y., editores, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pp. 3461--3474. Association for Computational Linguistics.

Mi, F.; Lin, X. & Faltings, B. (2020b). Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. Em *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, p. 408–413, New York, NY, USA. Association for Computing Machinery.

Michieli, U. & Zanuttigh, P. (2021). Knowledge distillation for incremental learning in semantic segmentation. *Computer Vision and Image Understanding*, 205:103167.

Mirzadeh, S.; Farajtabar, M.; Pascanu, R. & Ghasemzadeh, H. (2020). Understanding the role of training regimes in continual learning. Em Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F. & Lin, H., editores, *Advances in Neural Information Processing Systems*, volume 33, pp. 7308--7320. Curran Associates, Inc.

Monaikul, N.; Castellucci, G.; Filice, S. & Rokhlenko, O. (2021). Continual learning for named entity recognition. Em *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Movellan, J. R. (1991). Contrastive hebbian learning in the continuous hopfield model. Em Touretzky, D. S.; Elman, J. L.; Sejnowski, T. J. & Hinton, G. E., editores, *Connectionist Models*, pp. 10–17. Morgan Kaufmann.

Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. Em *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, p. 807–814, Madison, WI, USA. Omnipress.

Nguyen, C. V.; Li, Y.; Bui, T. D. & Turner, R. E. (2018). Variational continual learning. Em *International Conference on Learning Representations*.

Norvig, P. & Russell, S. (2004). Inteligencia artificial. *Editora Campus*, 20.

Oord, A. v. d.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Ostapenko, O.; Puscas, M.; Klein, T.; Jahnichen, P. & Nabi, M. (2019). Learning to remember: A synaptic plasticity driven framework for continual learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11321--11329.

Ostapenko, O.; Rodriguez, P.; Caccia, M. & Charlin, L. (2021). Continual learning via local module composition. *Advances in Neural Information Processing Systems*, 34.

Pan, P.; Swaroop, S.; Immer, A.; Eschenhagen, R.; Turner, R. E. & Khan, M. E. (2020). Continual deep learning by functional regularisation of memorable past. Em *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, volume 33, pp. 4453--4464. Curran Associates, Inc.

Pascoal, L. M. L.; do Nascimento, H. A. D.; Rosa, T. C.; Silva, E. Q. & Aleixo, E. L. (2020). Using string-comparison measures to improve and evaluate collaborative filtering recommender systems. Em *Bias and Social Aspects in Search and Recommendation: First International Workshop, BIAS 2020*, pp. 181--194.

Pennington, J.; Socher, R. & Manning, C. D. (2014). Glove: Global vectors for word representation. Em *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532--1543.

Perez-Rua, J.-M.; Zhu, X.; Hospedales, T. M. & Xiang, T. (2020). Incremental few-shot object detection. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13846--13855.

Petit, G.; Popescu, A.; Schindler, H.; Picard, D. & Delezoide, B. (2023). Fetril: Feature translation for exemplar-free class-incremental learning. Em *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3911–3920.

Pham, Q.; Liu, C. & Steven, H. (2022). Continual normalization: Rethinking batch normalization for online continual learning. Em *10th International Conference on Learning Representations, ICLR 2022, Virtual, April 25 - April 29, 2022, Conference Track Proceedings*.

Pham, T.; Tran, T.; Phung, D. & Venkatesh, S. (2016). Deepcare: A deep dynamic memory model for predictive medicine. Em *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 30--41. Springer.

PourKeshavarzi, M.; Zhao, G. & Sabokrou, M. (2022). Looking back on learned experiences for class/task incremental learning. Em *10th International Conference on Learning Representations, ICLR 2022, Virtual, April 25 - April 29, 2022, Conference Track Proceedings*.

Prabhu, A.; Torr, P. H. S. & Dokania, P. K. (2020). Gdumb: A simple approach that questions our progress in continual learning. Em Vedaldi, A.; Bischof, H.; Brox, T. & Frahm, J.-M., editores, *Computer Vision – ECCV 2020*, pp. 524--540, Cham. Springer International Publishing.

Qin, Q.; Hu, W.; Peng, H.; Zhao, D. & Liu, B. (2021). Bns: Building network structures dynamically for continual learning. *Advances in Neural Information Processing Systems*, 34.

Rajasegaran, J.; Khan, S.; Hayat, M.; Khan, F. S. & Shah, M. (2020). itaml: An incremental task-agnostic meta-learning approach. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13588--13597.

Ramesh, R. & Chaudhari, P. (2022). Model zoo: A growing brain that learns continually. Em *International Conference on Learning Representations*.

Rannen, A.; Aljundi, R.; Blaschko, M. B. & Tuytelaars, T. (2017). Encoder based lifelong learning. Em *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1320--1328.

Rebuffi, S.; Kolesnikov, A.; Sperl, G. & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. Em *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5533--5542. IEEE Computer Society.

Recht, B.; Fazel, M. & Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471--501.

Ritter, H.; Botev, A. & Barber, D. (2018). Online structured laplace approximations for overcoming catastrophic forgetting. Em Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N. & Garnett, R., editores, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 3742--3752.

Robins, A. V. (1993). Catastrophic forgetting in neural networks: the role of rehearsal mechanisms. Em *First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, ANNES '93, Dunedin, New Zealand, November 24-26, 1993*, pp. 65--68. IEEE.

Robins, A. V. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connect. Sci.*, 7(2):123--146.

Ronneberger, O.; Fischer, P. & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. Em *International Conference on Medical image computing and computer-assisted intervention*, pp. 234--241. Springer.

Rostami, M. (2021). Lifelong domain adaptation via consolidated internal distribution. *Advances in Neural Information Processing Systems*, 34.

Rostami, M.; Kolouri, S. & Pilly, P. K. (2019). Complementary learning for overcoming catastrophic forgetting using experience replay. Em *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3339--3345. International Joint Conferences on Artificial Intelligence Organization.

Roy, D.; Panda, P. & Roy, K. (2020). Tree-cnn: A hierarchical deep convolutional neural network for incremental learning. *Neural Networks*, 121:148--160.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. S.; Berg, A. C. & Li, F. (2015). Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211--252.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R. & Hadsell, R. (2016). Progressive neural networks. *CoRR*, abs/1606.04671.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D. & Lillicrap, T. P. (2016). Meta-learning with memory-augmented neural networks. Em Balcan, M. & Weinberger, K. Q., editores, *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 1842–1850. JMLR.org.

Sarfraz, F.; Arani, E. & Zonooz, B. (2023). Sparse coding in a dual memory system for lifelong learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9714--9722.

Schaefer, T. J. (1978). The complexity of satisfiability problems. Em *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, p. 216–226, New York, NY, USA. Association for Computing Machinery.

Schroff, F.; Kalenichenko, D. & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. Em *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 815--823. IEEE Computer Society.

Serrà, J.; Suris, D.; Miron, M. & Karatzoglou, A. (2018). Overcoming catastrophic forgetting with hard attention to the task. Em Dy, J. G. & Krause, A., editores, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4555--4564. PMLR.

Serre, D. (2002). *Elementary Theory*, pp. 1--14. Springer.

Shankar, S. & Sarawagi, S. (2018). Labeled memory networks for online model adaptation. Em McIlraith, S. A. & Weinberger, K. Q., editores, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 4034--4041. AAAI Press.

Shi, Y.; Zhou, K.; Liang, J.; Jiang, Z.; Feng, J.; Torr, P. H.; Bai, S. & Tan, V. Y. (2022). Mimicking the oracle: an initial phase decorrelation approach for class incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16722--16731.

Shim, D.; Mai, Z.; Jeong, J.; Sanner, S.; Kim, H. & Jang, J. (2021). Online class-incremental continual learning with adversarial shapley value. Em *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9630--9638.

Shin, H.; Lee, J. K.; Kim, J. & Kim, J. (2017). Continual learning with deep generative replay. Em Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S. & Garnett, R., editores, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Shmelkov, K.; Schmid, C. & Alahari, K. (2017). Incremental learning of object detectors without catastrophic forgetting. Em *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 3400--3409.

Simon, C.; Koniusz, P. & Harandi, M. (2021). On learning the geodesic path for incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1591--1600.

Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. Em *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Singh, P.; Verma, V. K.; Mazumder, P.; Carin, L. & Rai, P. (2020). Calibrating cnns for lifelong learning. Em Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. & Lin, H., editores, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Smith, J.; Hsu, Y.; Balloch, J.; Shen, Y.; Jin, H. & Kira, Z. (2021). Always be dreaming: A new approach for data-free class-incremental learning. *CoRR*, abs/2106.09701.

Sodhani, S.; Chandar, S. & Bengio, Y. (2020). Toward training recurrent neural networks for lifelong learning. *Neural Computation*, 32(1):1–35.

Sprechmann, P.; Jayakumar, S. M.; Rae, J. W.; Pritzel, A.; Badia, A. P.; Uria, B.; Vinyals, O.; Hassabis, D.; Pascanu, R. & Blundell, C. (2018). Memory-based parameter adaptation. Em *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Stanley, K. O. & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127.

Sun, S.; Calandriello, D.; Hu, H.; Li, A. & Titsias, M. (2022). Information-theoretic online memory selection for continual learning. Em *10th International Conference on Learning Representations, ICLR 2022, Virtual, April 25 - April 29, 2022, Conference Track Proceedings*.

Tang, S.; Chen, D.; Zhu, J.; Yu, S. & Ouyang, W. (2021). Layerwise optimization by gradient decomposition for continual learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9634--9643.

Tian, Z.; Shen, C.; Chen, H. & He, T. (2019). FCOS: Fully convolutional one-stage object detection. Em *Proc. Int. Conf. Computer Vision (ICCV)*.

Titsias, M. K.; Schwarz, J.; de G. Matthews, A. G.; Pascanu, R. & Teh, Y. W. (2020). Functional regularisation for continual learning with gaussian processes. Em *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Turing, A. (2004). *Intelligent Machinery (1948)*, pp. 395--432. Oxford University Press.

Turner, C. R.; Fuggetta, A.; Lavazza, L. & Wolf, A. L. (1999). A conceptual basis for feature engineering. *Journal of Systems and Software*, 49(1):3--15.

van de Ven, G. M.; Siegelmann, H. T. & Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1).

Van Hasselt, H.; Guez, A. & Silver, D. (2016). Deep reinforcement learning with double q-learning. Em *Thirtieth AAAI conference on artificial intelligence*.

Vapnik, V. (1991). Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4.

Velez, R. & Clune, J. (2017). Diffusion-based neuromodulation can eliminate catastrophic forgetting in simple neural networks. *PloS one*, 12(11):e0187736.

Verma, V. K.; Liang, K. J.; Mehta, N.; Rai, P. & Carin, L. (2021). Efficient feature transformations for discriminative and generative continual learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13865--13875.

Verwimp, E.; De Lange, M. & Tuytelaars, T. (2021). Rehearsal revealed: The limits and merits of revisiting samples in continual learning. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9385--9394.

von Oswald, J.; Henning, C.; Sacramento, J. & Grewe, B. F. (2020). Continual learning with hypernetworks. Em *International Conference on Learning Representations*.

Von Oswald, J.; Zhao, D.; Kobayashi, S.; Schug, S.; Caccia, M.; Zucchet, N. & Sacramento, J. (2021). Learning where to learn: Gradient sparsity in meta and continual learning. *Advances in Neural Information Processing Systems*, 34.

Wagner, G. P.; Pavlicev, M. & Cheverud, J. M. (2007). The road to modularity. *Nature Reviews Genetics*, 8(12):921--931.

Wang, L.; Zhang, X.; Yang, K.; Yu, L.; Li, C.; Hong, L.; Zhang, S.; Li, Z.; Zhong, Y. & Zhu, J. (2022a). Memory replay with data compression for continual learning. Em *10th International Conference on Learning Representations, ICLR 2022, Virtual, April 25 - April 29, 2022, Conference Track Proceedings*.

Wang, Y.; Huang, Z. & Hong, X. (2022b). S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. Em *Conference on Neural Information Processing Systems (NeurIPS)*.

Wang, Z.; Liu, L.; Duan, Y.; Kong, Y. & Tao, D. (2022c). Continual learning with lifelong vision transformer. Em *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 171–181.

Wang, Z.; Zhan, Z.; Gong, Y.; Yuan, G.; Niu, W.; Jian, T.; Ren, B.; Ioannidis, S.; Wang, Y. & Dy, J. (2022d). SparCL: Sparse continual learning on the edge. Em Oh, A. H.; Agarwal, A.; Belgrave, D. & Cho, K., editores, *Advances in Neural Information Processing Systems*.

Widrow, B.; Greenblatt, A.; Kim, Y. & Park, D. (2013). The no-prop algorithm: A new learning algorithm for multilayer neural networks. *Neural Networks*, 37:182--188.

Wu, C.; Herranz, L.; Liu, X.; wang, y.; van de Weijer, J. & Raducanu, B. (2018a). Memory replay gans: Learning to generate new categories without forgetting. Em Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N. & Garnett, R., editores, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Wu, G.; Gong, S. & Li, P. (2021). Striking a balance between stability and plasticity for class-incremental learning. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1124--1133.

Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y. & Fu, Y. (2019). Large scale incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y.; Zhang, Z. & Fu, Y. (2018b). Incremental classifier learning with generative adversarial networks. *arXiv preprint arXiv:1802.00853*.

Xiang, Y.; Fu, Y.; Ji, P. & Huang, H. (2019). Incremental learning using conditional adversarial networks. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6619--6628.

Xiong, F.; Liu, Z. & Yang, X. (2018). Overcoming catastrophic forgetting with self-adaptive identifiers. Em Cheng, L.; Leung, A. C. & Ozawa, S., editores, *Neural Information Processing - 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part III*, volume 11303 of *Lecture Notes in Computer Science*, pp. 497--505. Springer.

Xu, J. & Zhu, Z. (2018). Reinforced continual learning. Em Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N. & Garnett, R., editores, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 907--916.

Xu, Y.; Zhang, Y.; Guo, W.; Guo, H.; Tang, R. & Coates, M. (2020). *GraphSAIL: Graph Structure Aware Incremental Learning for Recommender Systems*, p. 2861–2868. Association for Computing Machinery, New York, NY, USA.

Xue, M.; Zhang, H.; Song, J. & Song, M. (2022). Meta-attention for vit-backed continual learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 150–159.

Yan, S.; Xie, J. & He, X. (2021). Der: Dynamically expandable representation for class incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3014--3023.

Yang, Y.; Yuan, H.; Li, X.; Lin, Z.; Torr, P. & Tao, D. (2023). Neural collapse inspired feature-classifier alignment for few-shot class-incremental learning. Em *The Eleventh International Conference on Learning Representations*.

Yang, Y.; Zhou, D.-W.; Zhan, D.-C.; Xiong, H. & Jiang, Y. (2019). Adaptive deep models for incremental learning: Considering capacity scalability and sustainability. Em *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 74–82, New York, NY, USA. Association for Computing Machinery.

Yegnanarayana, B. (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.

Yin, H.; Li, P. et al. (2021). Mitigating forgetting in online continual learning with neuron calibration. *Advances in Neural Information Processing Systems*, 34.

Yin, H.; Molchanov, P.; Alvarez, J. M.; Li, Z.; Mallya, A.; Hoiem, D.; Jha, N. K. & Kautz, J. (2020). Dreaming to distill: Data-free knowledge transfer via deepinversion. Em *The IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*.

Yoon, J.; Madaan, D.; Yang, E. & Hwang, S. J. (2022). Online coreset selection for rehearsal-based continual learning. Em *10th International Conference on Learning Representations, ICLR 2022, Virtual, April 25 - April 29, 2022, Conference Track Proceedings*.

Yoon, J.; Yang, E.; Lee, J. & Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. Em *International Conference on Learning Representations*.

Yoon, S. W.; Kim, D.-Y.; Seo, J. & Moon, J. (2020). XtarNet: Learning to extract task-adaptive representation for incremental few-shot learning. Em III, H. D. & Singh, A., editores, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10852--10860. PMLR.

Yu, L.; Twardowski, B.; Liu, X.; Herranz, L.; Wang, K.; Cheng, Y.; Jui, S. & Weijer, J. v. d. (2020). Semantic drift compensation for class-incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6982--6991.

Yu, Q.; Yang, Y.; Liu, F.; Song, Y.-Z.; Xiang, T. & Hospedales, T. M. (2017). Sketch-a-net: A deep neural network that beats humans. *International journal of computer vision*, 122(3):411--425.

Zacarias, A. S. & Alexandre, L. A. (2018a). Improving sena-cnn by automating task recognition. Em Yin, H.; Camacho, D.; Novais, P. & Tallón-Ballesteros, A. J., editores, *Intelligent Data Engineering and Automated Learning - IDEAL 2018 - 19th International Conference, Madrid, Spain, November 21-23, 2018, Proceedings, Part I*, volume 11314 of *Lecture Notes in Computer Science*, pp. 711--721. Springer.

Zacarias, A. S. & Alexandre, L. A. (2018b). Sena-cnn: Overcoming catastrophic forgetting in convolutional neural networks by selective network augmentation. Em Pancioni, L.; Schwenker, F. & Trentin, E., editores, *Artificial Neural Networks in Pattern Recognition - 8th IAPR TC3 Workshop, ANNPR 2018, Siena, Italy, September 19-21, 2018, Proceedings*, volume 11081 of *Lecture Notes in Computer Science*, pp. 102--112. Springer.

Zbontar, J.; Jing, L.; Misra, I.; LeCun, Y. & Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. Em *International Conference on Machine Learning*, pp. 12310--12320. PMLR.

Zeiler, M. D. & Fergus, R. (2014). Visualizing and understanding convolutional networks. Em Fleet, D.; Pajdla, T.; Schiele, B. & Tuytelaars, T., editores, *Computer Vision – ECCV 2014*, pp. 818--833, Cham. Springer International Publishing.

Zenke, F.; Poole, B. & Ganguli, S. (2017). Continual learning through synaptic intelligence. Em *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 3987–3995. JMLR.org.

Zhai, M.; Chen, L.; He, J.; Nawhal, M.; Tung, F. & Mori, G. (2020). Piggyback gan: Efficient lifelong learning for image conditioned generation. Em Vedaldi, A.; Bischof, H.; Brox, T. & Frahm, J.-M., editores, *Computer Vision – ECCV 2020*, pp. 397--413, Cham. Springer International Publishing.

Zhai, M.; Chen, L. & Mori, G. (2021). Hyper-lifelonggan: Scalable lifelong learning for image conditioned generation. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2246--2255.

Zhai, M.; Chen, L.; Tung, F.; He, J.; Nawhal, M. & Mori, G. (2019). Lifelong gan: Continual learning for conditional image generation. Em *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Zhang, Y.; Pfahringer, B.; Frank, E.; Bifet, A.; Lim, N. J. S. & Jia, A. (2022). A simple but strong baseline for online continual learning: Repeated augmented rehearsal. Em Oh, A. H.; Agarwal, A.; Belgrave, D. & Cho, K., editores, *Advances in Neural Information Processing Systems*.

Zheng, E.; Yu, Q.; Li, R.; Shi, P. & Haake, A. (2021). A continual learning framework for uncertainty-aware interactive image segmentation. Em *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6030--6038.

Zhou, D.-W.; Wang, Q.-W.; Ye, H.-J. & Zhan, D.-C. (2023). A model or 603 exemplars: Towards memory-efficient class-incremental learning. Em *The Eleventh International Conference on Learning Representations*.

Zhu, F.; Cheng, Z.; Zhang, X.-y. & Liu, C.-l. (2021a). Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34.

Zhu, F.; Zhang, X.-Y.; Wang, C.; Yin, F. & Liu, C.-L. (2021b). Prototype augmentation and self-supervision for incremental learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5871--5880.

Zurada, J. M. (1992). *Introduction to artificial neural systems*, volume 8. West publishing company St. Paul.