UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Lucas Augusto Mota de Alcantara

Aprendizagem Contínua para Classificação de Imagens

Recife

2024

Lucas Augusto Mota de Alcantara

Aprendizagem Contínua para Classificação de Imagens

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Mestre Acadêmico em Ciência da Computação.

Área de Concentração: Ciência da Computação

Orientador (a): Tsang Ing Ren

Recife

2024

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Alcantara, Lucas Augusto Mota de.
Aprendizagem Contínua para Classificação de Imagens / Lucas Augusto Mota de Alcantara. - Recife, 2024.
56f.: il.

Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação/PPGCC.
Orientação: Tsang Ing Ren.

1. Aprendizado Contínuo; 2. Esquecimento Catastrófico; 3. Pseudo Replay. I. Ren, Tsang Ing. II. Título.

UFPE-Centro de Ciencias Exatas e da Natureza

Lucas Augusto Mota de Alcantara

“Aprendizagem Contínua para Classificação de Imagens”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional

Aprovado em: 26/03/2024.

BANCA EXAMINADORA

_____

Prof. Dr. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE

_____

Prof. Dr. Luiz Eduardo Soares de Oliveira
Departamento de Informática /UFPR

_____

Prof. Dr. Tsang Ing Ren
Centro de Informática / UFPE
(**orientador**)

RESUMO

A habilidade de realizar Aprendizado Contínuo (*Continual Learning*) é crucial para o desenvolvimento de modelos de Inteligência Artificial capazes de adquirir e manter conhecimento ao longo do tempo sem esquecer informações anteriores. Isso representa um grande desafio técnico, dado que redes neurais são suscetíveis ao fenômeno de esquecimento catastrófico durante o processo de aprendizado de novas tarefas. Métodos basados na abordagem de *pseudo-replay* utilizam redes gerativas para criar amostras sintéticas de tarefas anteriores, que são então apresentadas ao modelo durante o aprendizado de novas tarefas com o intuito de reduzir o esquecimento. Nesta dissertação, exploramos melhorias no então método estado da arte baseado na abordagem de *pseudo-replay*, *Invariant Representation for Continual Learning* (IRCL). Utilizamos como modelo gerativo uma cVAE-GAN (*Conditional Variational Autoencoder Generative Adversarial Network*) e desacoplamos o seu treinamnto do restante da arquitetura, de forma a otimizar as diferentes partes da rede de forma independente. Além disso, utilizamos camadas convolucionais ao invés de camadas lineares. Os resultados experimentais alcançados demonstram melhorias de até 10 pontos percentuais na Acurácia Média e de até 8 pontos na média do *Backward Transfer*, superando o estado da arte nos conjuntos de dados *Split MNIST* e *Split FashionMNIST*.

Palavras-chaves: Aprendizado Contínuo, Esquecimento Catastrófico, Pseudo Replay.

ABSTRACT

Continual Learning is the concept of having a model able to sequentially learn to solve new tasks without losing the ability to solve previous tasks. Achieving this is challenging because neural networks usually suffer from catastrophic forgetting of the preceding tasks when they are learning new ones. To handle this issue, pseudo-replay approaches leverages the performance of generative networks using them to generate samples related to past data to serve as input to the model when it is learning new tasks. In this work, we propose an improved architecture and training strategy based on the state-of-the-art pseudo-replay IRCL method. We use a cVAE-GAN as the generative model and train it decoupled from the other components of the architecture. Also, we make use of convolutional layers for the architecture components instead of linear ones. Our experimental results show that the proposed method outperforms the state-of-the-art IRCL method by up to 10% in Average Accuracy and up to 8.3% in Average Backward Transfer on both Split MNIST and Split FashionMNIST datasets.

Keywords: Continual Learning, Catastrophic Forgetting, Pseudo-Replay.

LISTA DE FIGURAS

LISTA DE TABELAS

super

SUMÁRIO

1 INTRODUÇÃO

As redes neurais artificiais são uma subárea da aprendizagem de máquina (*machine learning*) que revolucionou a forma como resolvemos problemas complexos em diversas aplicações, desde processamento de imagens e áudio até processamento de linguagem natural e robótica. Em essência, redes neurais são sistemas computacionais inspirados na estrutura e funcionamento do cérebro humano, capazes de aprender com dados a reconhecer padrões, fazer previsões e gerar dados. Isso as torna uma ferramenta poderosa na resolução de problemas do mundo real que podem ser muito complexos ou difíceis de serem resolvidos com algoritmos tradicionais.

Embora o conceito de redes neurais exista desde a década de 1940 (MCCULLOCH; PITTS, 1943), seu verdadeiro potencial só começou a ser explorado com o surgimento da computação moderna, que trouxe a capacidade de geração, armazenamento e processamento de grandes volumes de dados. Isso permitiu treinar redes neurais profundas, que possuem múltiplas camadas de processamento e que são capazes de aprender características complexas e abstratas dos dados (CIREsAN et al., 2011) (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) (CIREGAN; MEIER; SCHMIDHUBER, 2012), o que impulsionou a subárea do aprendizado de máquina conhecida como aprendizado profundo (*deep learning*). Desde então, esse é um campo de pesquisa em rápida evolução, com novas arquiteturas, algoritmos e aplicações sendo desenvolvidas constantemente.

Como consequência, nos últimos anos houve um avanço significativo no desempenho das redes profundas para solucionar problemas de visão computacional, tais como classificação de imagens e reconhecimento de objetos, atividades onde, em muitos casos, as redes conseguem superar o desempenho humano (HE et al., 2015). Por exemplo, em tarefas de detecção de objetos, os modelos de aprendizado profundo são capazes de detectar e localizar com precisão múltiplos objetos em uma imagem, mesmo em cenas complexas, em uma fração de segundo. Essa velocidade é muito superior à capacidade humana de processar informações visuais, tornando esses modelos ideais para automatizar tarefas de processamento de imagens.

No entanto, esses modelos ainda têm limitações em comparação ao desempenho humano em alguns aspectos. Por exemplo, os humanos têm a capacidade de raciocinar sobre o conteúdo de uma imagem e, então, entender o contexto e o significado de uma forma que ainda não é totalmente replicada pelos modelos de aprendizagem profunda. Humanos também podem

aprender a partir de poucos exemplos e generalizar para novas situações não vistas, enquanto os modelos profundos frequentemente requerem dados rotulados de qualidade e em grande quantidade para alcançar um alto desempenho.

Outra limitação significativa é que a maioria das abordagens usadas para resolver esses problemas assume que os dados de treinamento são independentes e identicamente distribuídos (i.i.d.). Isso significa que o desempenho desses modelos é muito reduzido em cenários em que os dados de treinamento não estão de acordo com essa suposição, como quando o conhecimento precisa ser ampliado a partir de dados recebidos de forma sequencial, como ilustrado na Figura 1.

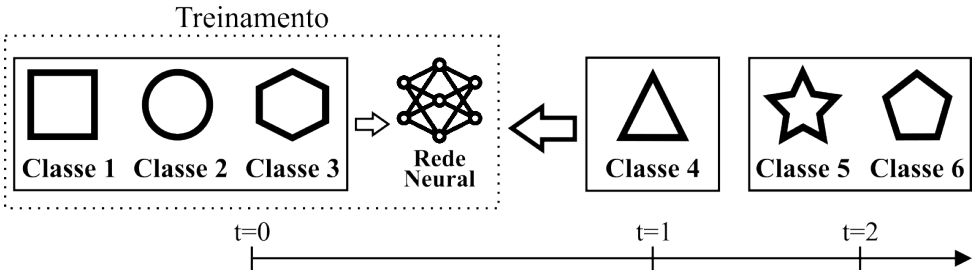


Figura 1 – Aprendizado com dados acessados sequencialmente.

Por outro lado, os seres humanos têm a capacidade de aprender a realizar novas tarefas enquanto acumulam o novo conhecimento com aquele adquirido no passado. Além disso, durante o aprendizado das novas tarefas utilizam a experiência acumulada até então, como quando primeiro aprendemos a falar e, posteriormente, a ler. Essa habilidade, chamada de aprendizagem contínua (*continual learning* - CL) ou aprendizagem incremental (*incremental learning* - IL), é uma capacidade que sistemas inteligentes devem possuir para lidar com problemas do mundo real, uma vez que eles evoluem constantemente e padrões que não eram conhecidos antes precisam ser aprendidos por um sistema já existente.

1.1 MOTIVAÇÃO

Apesar de se inspirar no funcionamento do cérebro humano, a abordagem tradicionalmente usada para ensinar novas tarefas para redes neurais consiste em periodicamente descartar o modelo atual (ou parte dele) e treinar um novo usando todos os dados coletados até o momento, em um processo chamado de *Continuous Training* (SYMEONIDIS et al., 2022), processo ilustrado na Figura 2, algo que nem sempre é viável devido a restrições de tempo ou de capacidade computacional, além de não reproduzir o aprendizado humano.

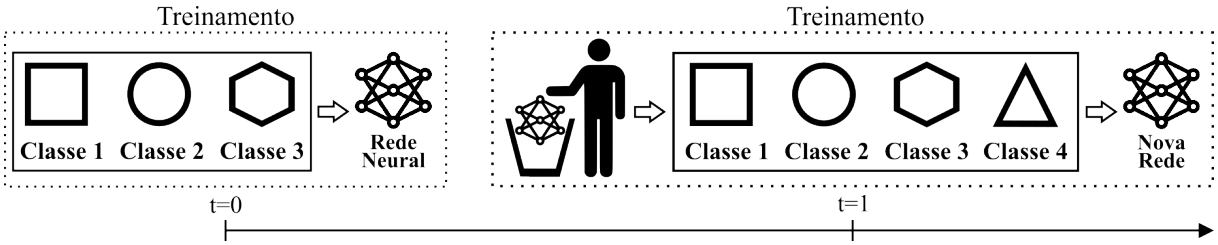


Figura 2 – *Continuous Training* - A cada vez que os dados são atualizados, o modelo existente é descartado e um novo modelo é treinado.

Para resolver isso, métodos de aprendizagem contínua têm o objetivo de evitar a perda de conhecimentos adquiridos no passado durante o aprendizado de novas tarefas, permitindo que os modelos retenham o conhecimento obtido a partir de dados acessados sequencialmente, sem a necessidade de treinar um novo modelo sempre que novas tarefas precisam ser aprendidas.

Consequentemente, adicionar a capacidade de aprendizado contínuo a sistemas inteligentes também pode trazer benefícios significativos do ponto de vista da sustentabilidade (COSSU; ZIOSI; LOMONACO, 2021), uma vez que possibilitaria a redução na quantidade de recursos computacionais necessários para treinar modelos em novos dados. Como esses modelos poderiam se aproveitar de um conhecimento já adquirido, eles exigiriam menos tempo e energia para aprender novas informações, reduzindo a pegada de carbono do treinamento dos modelos, tornando-os mais eficientes e sustentáveis.

Por exemplo, o treinamento do modelo CoAtNet (DAI et al., 2021), uma rede neural profunda moderna para classificação de imagens, com o dataset JFT-3B (ZHAI et al., 2021) resulta em um custo estimado de cerca de 83.000kg CO2eq (LI et al., 2022). Em comparação, o custo anual médio gerado por um ser humano é de aproximadamente 5.000kg CO2eq (STRUBELL; GANESH; MCCALLUM, 2020). Devido ao alto custo de treinamento, é desejável encontrar alternativas que possibilitem a inclusão de novas classes de maneira mais eficiente, evitando o custo do treinamento completo do modelo.

Entretanto, construir um sistema com a capacidade de aprender continuamente é algo desafiador, porque as redes neurais geralmente otimizam seus pesos com base nos dados mais recentes, o que pode fazer com que elas esqueçam o conhecimento relacionado a dados passados, um fenômeno conhecido como esquecimento catastrófico (*catastrophic forgetting*) ou interferência catastrófica (*catastrophic interference*).

Uma maneira comum de mitigar esse problema é armazenar os dados referentes ao conhecimento já adquirido pelo modelo para que sejam utilizados posteriormente para atualizá-lo quando necessário, mas sem descartá-lo por completo, uma abordagem conhecida como *rehe-*

arsal ou *replay*. No entanto, esses métodos tem a desvantagem de exigir muito espaço de armazenamento, o que pode ser impraticável em muitas situações.

Por outro lado, outros métodos buscam mitigar o efeito de esquecimento catastrófico sem depender do armazenamento dos dados através da geração de dados artificiais que representem os dados vistos no passado, uma abordagem conhecida como *pseudo-rehearsal* ou *pseudo-replay*. No entanto, esse é um campo de pesquisa que ainda está em estágio inicial na área de visão computacional com modelos profundos, de forma que os métodos propostos até o momento apresentam um desempenho limitado, que ainda não é adequado para aplicações práticas.

1.2 OBJETIVOS

Este trabalho tem como objetivo desenvolver um novo método que aprimore o desempenho de redes neurais treinadas continuamente, especificamente no contexto de classificação de imagens com aprendizado incremental de classes, utilizando a abordagem de *pseudo-rehearsal*. A proposta é melhorar o atual método estado-da-arte, IRCL (SOKAR; MOCANU; PECHENIZKIY, 2021), a partir da ideia de que, em métodos baseados em *pseudo-rehearsal*, o desempenho é limitado pela qualidade dos dados gerados (SHIN et al., 2017).

A partir dessa ideia foram definidas três hipóteses visando melhorar a qualidade das imagens geradas. A primeira delas consiste em modificar a arquitetura do IRCL para desacoplar o treinamento do modelo gerador dos demais módulos. A segunda hipótese propõe a implementação de um cVAE-GAN, através da adição de um discriminador após o decodificador presente na arquitetura original. Por fim, a terceira hipótese é a utilização de camadas convolucionais em alguns módulos da arquitetura, visando aprimorar a extração de características e, consequentemente, a geração de imagens.

O trabalho realizado resultou na seguinte publicação:

Convolutional Decoupled cVAE-GANs for Pseudo-Replay Based Continual Learning, L. A. M. De Alcantara, J. I. S. Da Silva, M. L. P. C. Silva, S. C. S. Machado and T. I. Ren, 2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI), Macao, China, 2022, pp. 585-590, doi: 10.1109/ICTAI56018.2022.00092.

1.3 ESTRUTURA DA DISSERTAÇÃO

Os capítulos desta dissertação estão divididos da seguinte forma:

- **Capítulo 2:** Apresentação dos conceitos fundamentais para o entendimento do trabalho.
- **Capítulo 3:** Histórico do campo de aprendizado contínuo, incluindo os trabalhos relacionados mais relevantes e uma explicação do então método estado da arte para classificação de imagens, baseado na abordagem de *pseudo-rehearsal*.
- **Capítulo 4:** Descrição detalhada do método proposto neste trabalho.
- **Capítulo 5:** Apresentação dos experimentos realizados e os resultados obtidos com a aplicação do método proposto.
- **Capítulo 6:** Conclusões a respeito do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo apresentar os conceitos fundamentais necessários para a compreensão do trabalho. Nele serão abordados temas como redes neurais e seus diferentes tipos de camadas, *autoencoders* e suas variações, além de conceitos essenciais para o entendimento de aprendizado contínuo, como tarefas, cenários e abordagens existentes.

2.1 REDES NEURAIS

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados pela estrutura e funcionamento do cérebro humano. Assim como o cérebro é composto por neurônios interconectados que trabalham conjuntamente para realizar tarefas complexas, as RNAs são formadas por neurônios artificiais, ilustrados na Figura 3, também conhecidos como nós, que se organizam em camadas para processar e interpretar dados (HAYKIN, 2009). Cada nó realiza cálculos sobre os dados de entrada, transformando esses dados em uma saída.

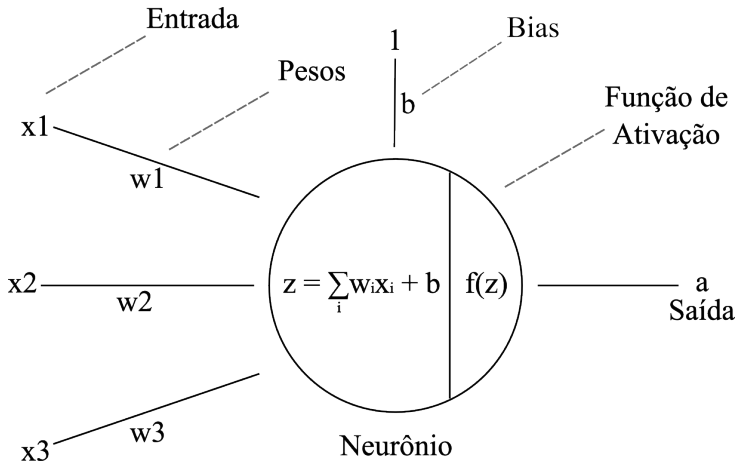


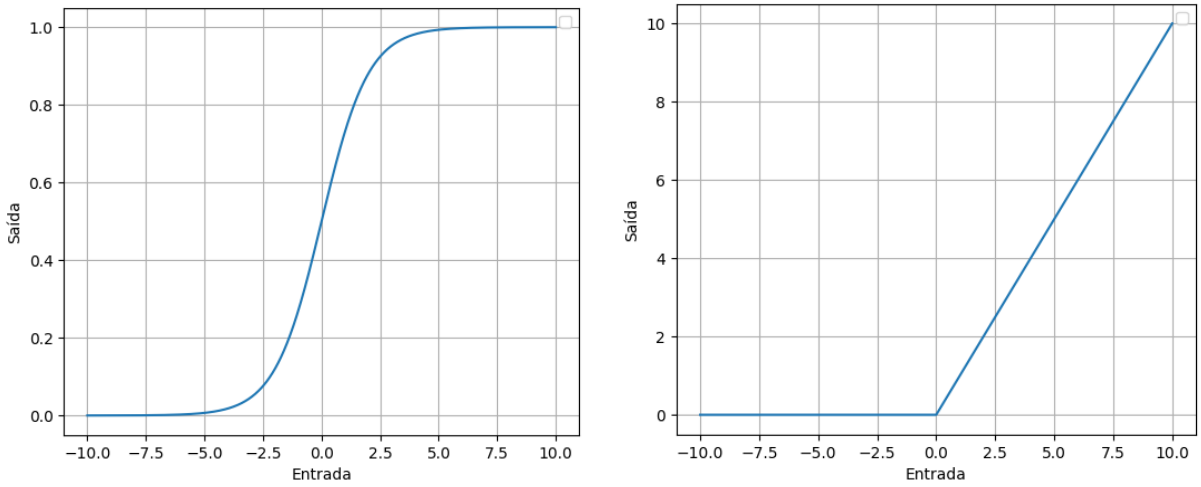
Figura 3 – Ilustração de um neurônio artificial

Isso confere às redes a habilidade de aprender a partir de dados, adaptando-se a novas entradas e aprimorando sua capacidade de previsão. Essas características as tornam ferramentas poderosas para resolver uma grande variedade de aplicações, como reconhecimento de imagem, processamento de linguagem natural e previsão de séries temporais.

Cada nó em uma Rede Neural Artificial (RNA) está associado a um conjunto de pesos, que determinam a influência das entradas no cálculo da saída. Além dos pesos, cada nó pode incluir um termo adicional conhecido como *bias*. Os pesos são utilizados para gerar uma combinação linear dos dados de entrada. Em seguida, o resultado dessa combinação é somado ao *bias*

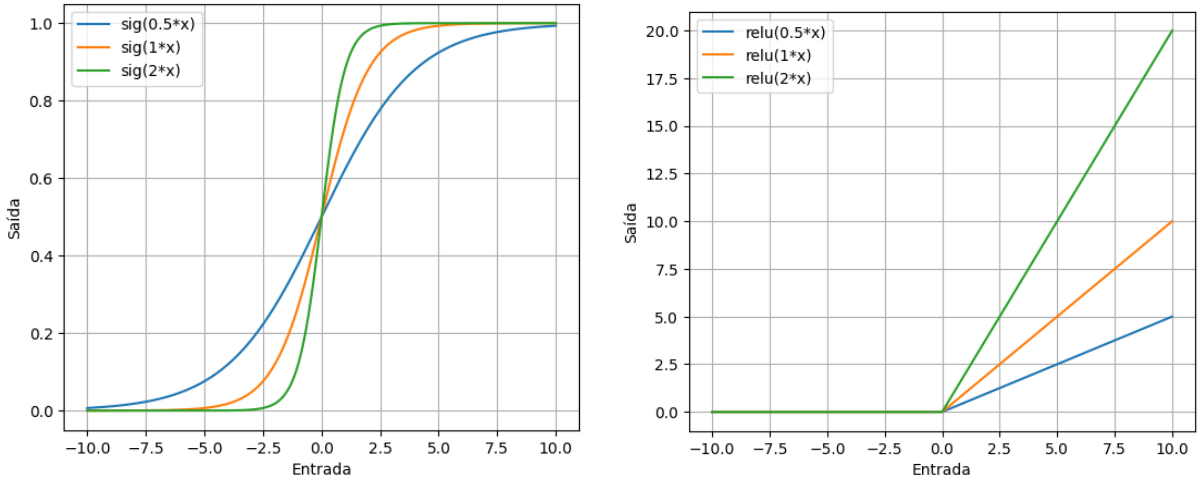
e aplicado a uma função não linear, chamada de função de ativação. A função de ativação introduz não-linearidade ao neurônio, permitindo que ele aprenda relações não lineares entre as entradas e a saída, de tal forma que, sem uma função de ativação não linear, o neurônio seria limitado a modelar apenas relações lineares (ZHANG et al., 2021), o que não seria suficiente para muitos problemas práticos.

A Figura 4 ilustra duas funções de ativação comumente utilizadas em redes neurais: Sigmoid e ReLU (Rectified Linear Unit). A função Sigmoid, definida como $f(x) = \frac{1}{1+e^{-x}}$, mapeia qualquer valor de entrada para um intervalo entre 0 e 1, sendo útil em tarefas de classificação binária. Já a função ReLU é definida como $f(x) = \max(0, x)$ e é amplamente utilizada devido à sua simplicidade e eficiência computacional.



(a) Sigmoid (b) ReLU

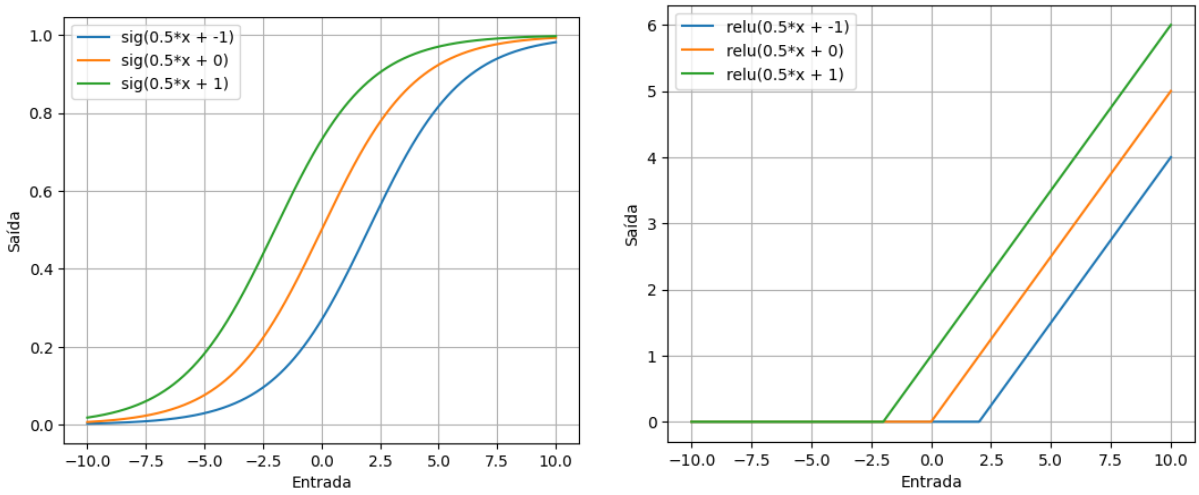
Figura 4 – Exemplos de funções de ativação



(a) Sigmoid (b) ReLU

Figura 5 – Exemplos de funções de ativação com diferentes pesos

O papel dos pesos pode ser interpretado como a definição da inclinação da função de ativação, como ilustrado na Figura 5 para as funções Sigmoid e ReLU. O *bias* funciona como um peso extra, mas com uma entrada constante de valor 1, permitindo que a função de ativação de cada nó seja deslocada para a esquerda ou para a direita permitindo que o neurônio produza uma saída diferente de 0 para uma entrada nula e, consequentemente, proporcionando maior flexibilidade e capacidade da rede para modelar dados. O efeito causado pelo *bias* na saída das funções de ativação Sigmoid e ReLU é ilustrado na Figura 6.



(a) Sigmoid (b) ReLU

Figura 6 – Exemplos de funções de ativação com diferentes valores de *bias*

Tanto os pesos quanto o *bias* são ajustados durante o processo de aprendizado da rede, através de um processo iterativo conhecido como treinamento. Nesse processo, a rede ajusta os pesos e os *bias* de seus nós com o objetivo de minimizar a diferença entre as saídas produzidas pela rede e as saídas desejadas. Ao longo de várias iterações de treinamento, a rede neural aprende a mapear as entradas às saídas corretas, melhorando a sua capacidade de previsão e aumentando a precisão e eficácia do modelo final.

2.2 CAMADAS

Os neurônios que compõem as redes neurais artificiais são interconectados e organizados em camadas, que podem ser divididas em camadas de entrada, camadas escondidas e camadas de saída (HAYKIN, 2009), e ainda em dois tipos básicos: camadas lineares e camadas convolucionais (ZHANG et al., 2021). As camadas de entrada são responsáveis por receber os dados e repassá-los para as camadas escondidas onde será feito o processamento. Cada nó desta

camada corresponde a um atributo ou variável dos dados e recebe um valor numérico que o representa. Em outras palavras, cada nó na camada de entrada corresponde a uma variável de entrada.

As camadas escondidas são as camadas entre a camada de entrada e a camada de saída e são responsáveis por extrair de características dos dados de entrada. Na primeira camada escondida, que é diretamente conectada à camada de entrada, os neurônios identificam características de baixo nível nos dados, que podem incluir, por exemplo, linhas ou bordas em uma imagem, ou certos fonemas em um sinal de áudio.

À medida que as informações passam para as camadas escondidas subsequentes, a rede neural começa a reconhecer padrões mais complexos e abstratos, usando as características de baixo nível identificadas pelas camadas anteriores. Isso é possível porque cada camada escondida recebe como entrada a saída da camada anterior, permitindo que a rede construa uma hierarquia de características. Por exemplo, em uma rede neural treinada para o reconhecimento de imagens, as primeiras camadas escondidas podem detectar bordas, enquanto camadas subsequentes podem começar a reconhecer formas mais complexas, como texturas ou objetos.

Assim, as camadas escondidas permitem que a rede neural transforme os dados brutos de entrada em uma representação mais útil e informativa, que será usada pela camada de saída para gerar o resultado desejado. Esta habilidade de aprender representações hierárquicas e abstratas dos dados é uma das principais razões pela qual as redes neurais são tão eficazes em uma ampla gama de tarefas de aprendizado de máquina. Essas camadas são chamadas de "escondidas" porque suas saídas não são diretamente observáveis, sendo utilizadas apenas internamente pela rede.

Por fim, as camadas de saída são responsáveis por gerar as saídas finais da rede neural. Dependendo da natureza do problema, a camada de saída pode ser composta de um ou vários nós. Por exemplo, em um problema de classificação binária, a camada de saída terá um nó, enquanto em um problema de classificação multiclasse, haverá um nó para cada classe.

2.2.1 Lineares

As camadas lineares, também conhecidas como camadas totalmente conectadas (*fully connected*) ou densas, são compostas por um conjunto de neurônios artificiais. Cada neurônio desta camada é conectado a todos os neurônios da camada anterior e da camada seguinte,

daí o termo "totalmente conectado" (HAYKIN, 2009). A Figura 7 ilustra uma rede neural com uma camada escondida do tipo linear.

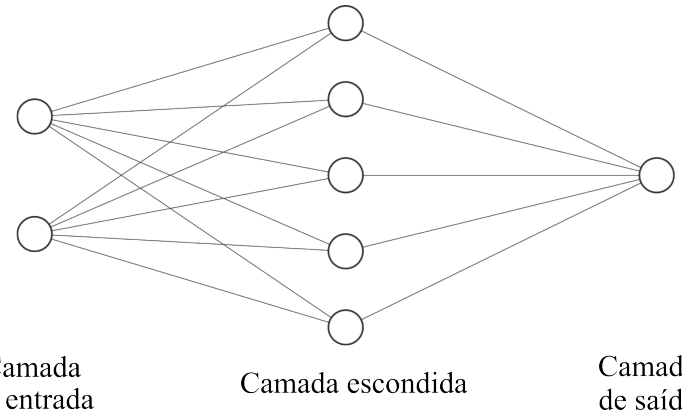


Figura 7 – Ilustração de rede neural com uma camada linear escondida

A função de uma camada linear é aplicar uma transformação linear aos dados de entrada, isso é feito multiplicando a matriz de entrada pelos pesos da camada e adicionando um vetor de *bias* (ZHANG et al., 2021). O resultado dessa operação é uma nova matriz que representa os dados transformados. A operação pode ser expressa como:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b \tag{2.1}$$

Onde X é a matriz de entrada, w é a matriz de pesos, b é o vetor de *bias* e y é a matriz resultante. Após a transformação linear, é aplicada uma função de ativação aos dados transformados.

Durante o treinamento, os pesos e os *bias* da camada linear são ajustados usando um algoritmo de otimização, como o gradiente descendente, com base no *feedback* do erro de saída da rede. O objetivo é minimizar a função de perda, que mede a diferença entre as previsões da rede e os valores reais (rótulos).

2.2.2 Convolucionais

Camadas convolucionais são os componentes fundamentais das redes neurais convolucionais (CNNs) (LECUN et al., 1989), que são um tipo de rede neural artificial projetada para processar e analisar dados com uma estrutura espacial, como imagens ou sinais de áudio (ZHANG et al., 2021). A Figura 8 ilustra uma rede convolucional.

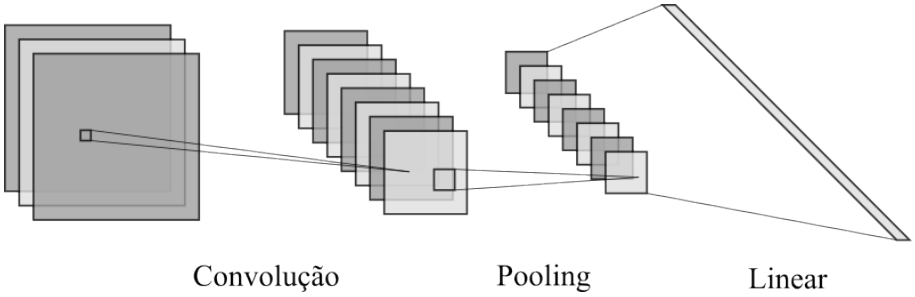


Figura 8 – Ilustração de uma rede neural com camadas convolucionais

As camadas convolucionais são mais adequadas para tarefas de visão computacional do que as camadas lineares devido à sua capacidade de capturar informações espaciais nas imagens. Ao contrário das camadas lineares, que processam as entradas como um vetor unidimensional, as camadas convolucionais consideram a estrutura espacial das entradas, o que ajuda na detecção padrões e características locais nos dados, como bordas, texturas e formas.

A camada convolucional usa filtros (também conhecidos como *kernels*), que são pequenas matrizes com valores ajustáveis (parâmetros). Esses filtros são aplicados aos dados de entrada por meio de uma operação chamada convolução, que é o processo de deslizar o filtro sobre a imagem (ou dados de entrada) e realizar uma multiplicação ponto a ponto (produto escalar) entre os valores do filtro e os valores da imagem que estão sob o filtro. Em seguida, os resultados dessas multiplicações são somados, gerando um único valor. Esse valor é armazenado em uma nova matriz chamada mapa de características (*feature map*) (ZHANG et al., 2021).

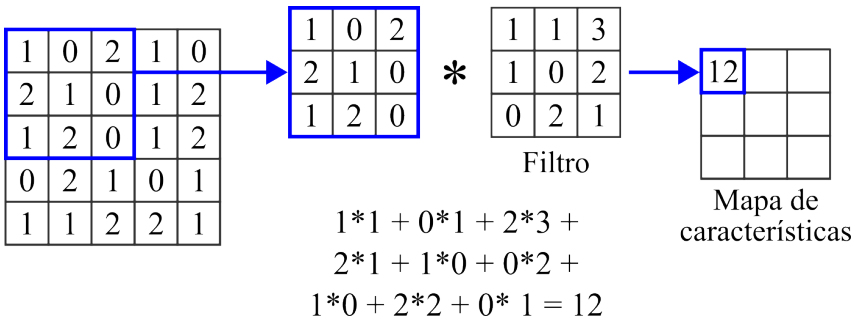


Figura 9 – Exemplo da operação de convolução

A distância que o filtro se move ao deslizar pela imagem no processo de convolução é determinada pelo hiperparâmetro tamanho do passo (*stride*) (ZHANG et al., 2021). Um passo maior significa que o filtro se move mais rapidamente pela imagem, resultando em um mapa de características menor. A Figura 10 ilustra o efeito do tamanho do passo na operação de convolução utilizando o mesmo filtro da Figura 9.

Outro hiperparâmetro é o preenchimento (*padding*), que adiciona *pixels* extras ao redor

(a) Passo = 1 (b) Passo = 2

Figura 10 – Exemplo de operação de convolução com diferentes tamanhos de passo

da borda da imagem para permitir que o filtro seja aplicado nas bordas da imagem e manter o tamanho da imagem após a convolução (ZHANG et al., 2021). A Figura 11 ilustra a aplicação de convolução com preenchimento de *pixels* com valor zero (*zero-padding*), com o mesmo filtro da Figura 9.

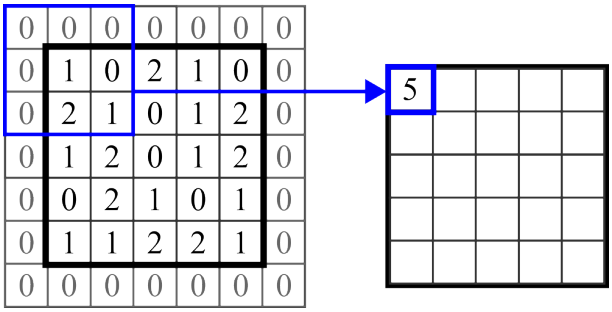


Figura 11 – Ilustração do efeito da aplicação de preenchimento

Após a convolução, uma função de ativação é aplicada a cada valor do mapa de características para introduzir não-linearidade ao modelo, permitindo que a rede aprenda relações mais complexas (ZHANG et al., 2021).

Embora não seja uma parte obrigatória das camadas convolucionais, o agrupamento (*pooling*) é frequentemente usado após a convolução para reduzir a dimensionalidade do mapa de características (ZHANG et al., 2021). O agrupamento oferece benefícios como a redução da sensibilidade a pequenas variações espaciais e a diminuição do número de parâmetros, o que ajuda a evitar o sobreajuste (*overfitting*) e melhora a eficiência computacional. A Figura 12 ilustra a operação de *max-pooling*, que funciona selecionando o valor máximo em cada região de agrupamento. Além do *max-pooling*, outra forma comum de agrupar valores do mapa de características é através da seleção do valor médio (*average-pooling)* da região.

As camadas convolucionais são organizadas sequencialmente em uma CNN, permitindo que a rede aprenda representações hierárquicas dos dados. As primeiras camadas costumam capturar características mais simples, como retas e curvas, enquanto as camadas subsequentes se concentram em características mais complexas, como texturas.

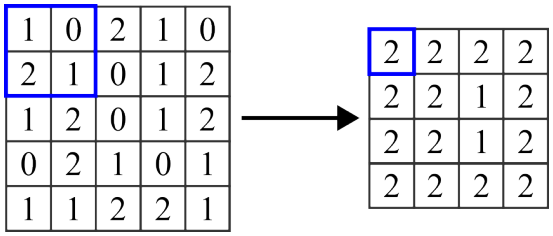


Figura 12 – Exemplo da operação de *Max-Pooling*

2.3 *AUTOENCODERS*

Os *autoencoders* (AE) são redes neurais que têm como objetivo principal aprender representações compactas e eficientes dos dados, frequentemente empregadas na redução de dimensionalidade e geração de novos exemplos que possuem similaridade com os dados treinados (**??**). A importância da redução de dimensionalidade reside em sua capacidade de tornar o processamento de dados mais eficiente e de destacar as características mais significativas. AEs são compostos por duas partes principais: um codificador (*encoder*) e um decodificador (*decoder*), como ilustrado na Figura 13.

A função do codificador é transformar os dados de entrada em uma representação de dimensão menor, chamada de representação latente, através de uma série de camadas que gradualmente reduzem a dimensionalidade dos dados de entrada.

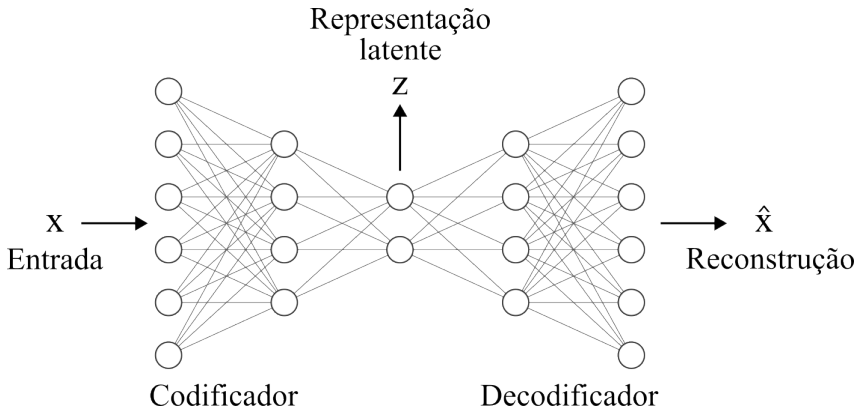


Figura 13 – Ilustração de um autoencoder tradicional

O decodificador, por sua vez, reconstrói os dados de entrada a partir da representação latente. A arquitetura do decodificador é geralmente simétrica à do codificador, revertendo o processo de redução e expandindo a dimensionalidade do código latente até atingir a dimensão original dos dados de entrada.

Os *autoencoders* são treinados para minimizar a diferença entre os dados de entrada e a reconstrução gerada pelo decodificador. Uma função de perda, como o erro quadrático médio,

pode ser usada para medir essa diferença.

2.3.1 *Autoencoders* Variacionais

Os *autoencoders* variacionais (VAEs) (KINGMA; WELLING, 2013) são uma extensão dos *autoencoders* tradicionais onde, em vez de aprender apenas uma representação latente para cada exemplo de entrada, os VAEs aprendem também a média e a variância de uma distribuição de probabilidade a partir da qual a representação latente pode ser amostrada. Essa abordagem permite que o VAE gere novos exemplos plausíveis a partir do espaço latente.

O treinamento de VAEs envolve a otimização de uma função de perda composta por duas partes: uma que mede a reconstrução (semelhante aos AEs) e outra que mede a divergência de *Kullback-Leibler* (KL) entre a distribuição de probabilidade aprendida e uma distribuição pré-definida (geralmente uma distribuição normal). A divergência KL atua como um regularizador, incentivando o espaço latente a seguir a distribuição definida.

2.3.2 *Autoencoders* Variacionais Condicionais

Autoencoders variacionais condicionais (CVAEs) (SOHN; LEE; YAN, 2015) são uma variação dos *autoencoders* variacionais que incorporam informações adicionais, como rótulos ou outras variáveis contextuais, no processo de codificação e decodificação. Essa abordagem permite que a rede aprenda representações que são condicionadas à informação adicional, o que torna os CVAEs mais flexíveis em tarefas como geração de dados condicionados.

No treinamento de CVAEs, a informação adicional é fornecida como entrada para o codificador ou para o decodificador, juntamente com os dados de entrada. A função de perda é semelhante à dos *autoencoders* variacionais, mas é calculada com base na reconstrução condicionada à informação adicional.

2.3.3 VAE-GANs

Os VAE-GANs (LARSEN et al., 2016) combinam elementos dos *autoencoders* variacionais e das redes generativas adversariais (GANs) (GOODFELLOW et al., 2020). GANs são uma classe de modelos generativos que consistem em duas redes neurais, um gerador e um discriminador, que competem entre si em um jogo de soma zero. O gerador tenta criar exemplos realistas,

enquanto o discriminador avalia a qualidade desses exemplos, tentando distinguir entre exemplos reais e gerados. Essa competição leva a um gerador capaz de criar exemplos altamente realistas.

VAE-GANs aproveitam as vantagens dos VAEs e GANs, integrando o processo de geração de VAEs com a estrutura adversarial das GANs, formando um modelo composto por três componentes: codificador, decodificador e discriminador, conforme ilustrado na Figura 14.

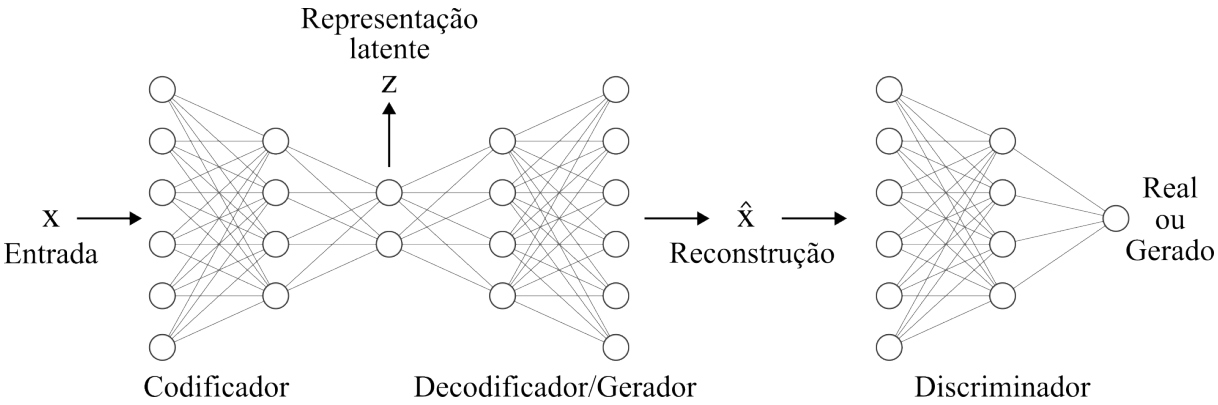


Figura 14 – Ilustração de um VAE-GAN

Semelhante ao VAE tradicional, o codificador aprende a média e a variância de uma distribuição de probabilidade a partir da qual o código latente é amostrado. Já o decodificador atua como o gerador de uma GAN, reconstruindo os dados de entrada a partir da representação latente e tentando gerar exemplos realistas. O discriminador é treinado para distinguir entre exemplos reais e gerados, avaliando a qualidade das reconstruções.

O treinamento dos VAE-GANs envolve a otimização de uma função de perda que combina os objetivos de reconstrução e adversarial. A parte de reconstrução é semelhante à dos VAEs, medindo a diferença entre os dados de entrada e a reconstrução gerada pelo decodificador e a divergência KL. A parte adversarial é semelhante à das GANs, com o decodificador (gerador) tentando enganar o discriminador e o discriminador tentando identificar corretamente os exemplos gerados.

2.4 TAREFAS EM APRENDIZADO CONTÍNUO

Uma tarefa é uma unidade de aprendizado ou um objetivo que um sistema de aprendizado de máquina deve realizar. Em geral, cada tarefa é definida por um conjunto de dados específicos e possivelmente diferentes características, como uma função de perda própria e um conjunto de métricas de desempenho associadas.

Em um cenário de aprendizado incremental, várias tarefas podem ser apresentadas ao modelo sequencialmente, e ele deve ser capaz de aprender cada uma delas sem esquecer completamente o que aprendeu anteriormente. Por exemplo, em uma tarefa de classificação de imagens, cada tarefa pode envolver a classificação de um conjunto diferente de objetos, como carros, flores ou animais. Além disso, espera-se que o modelo seja capaz de utilizar o conhecimento adquirido ao longo do tempo no processo de aprendizado das novas tarefas, combinando as informações relevantes das tarefas passadas com as informações das novas.

2.5 CENÁRIOS DE PROBLEMAS DE APRENDIZADO CONTÍNUO

Um cenário é um conjunto de condições específicas que definem como as tarefas serão apresentadas ao modelo ao longo do tempo, além de quais informações estão disponíveis durante o treinamento e inferência. Os diferentes cenários podem variar em termos de como as tarefas são organizadas e apresentadas, de forma que as técnicas de aprendizado desenvolvidas para cada cenário podem ser diferentes. Os cenários mais comuns são os de tarefa incremental, domínio incremental ou classe incremental (VEN; TOLIAS, 2019), onde cada um deles têm suas próprias características e desafios específicos. O entendimento e escolha do cenário adequado para cada aplicação é fundamental para obter bons resultados em implementações de aprendizado contínuo.

2.5.1 Tarefa Incremental

O cenário de tarefa incremental é um dos cenários de aprendizado contínuo em que o algoritmo deve aprender um conjunto de tarefas distintas de forma incremental, uma após a outra, de forma que sempre está claro para o algoritmo, tanto no treinamento quanto na inferência, qual tarefa deve ser realizada. Isso pode ser feito de forma explícita, fornecendo identificadores de tarefa ou tornando as tarefas claramente distinguíveis entre si.

O papel do identificador da tarefa é distinguir as tarefas que o modelo deve resolver. Ele é importante para garantir que o modelo possa separar as informações relacionadas a cada tarefa e evitar de misturá-las durante o aprendizado. Em outras palavras, é necessário que o modelo saiba em que tarefa está trabalhando no momento e que possa ajustar seus parâmetros de acordo com ela.

Uma das principais dificuldades desse cenário é encontrar maneiras eficazes de compartilhar

representações aprendidas entre tarefas, otimizando o desempenho e a complexidade computacional. Uma abordagem comum é treinar modelos com componentes específicos para cada tarefa, como as camadas de saída, ou até mesmo ter uma rede neural completamente separada para cada tarefa, o que elimina o risco de esquecimento catastrófico. No entanto, isso pode resultar em um grande número de parâmetros e pode não ser escalável em um grande número de tarefas.

Exemplo do cenário de tarefa incremental na vida real seria aprender a tocar diferentes instrumentos musicais, já que, em geral, instrumentos diferentes são bem distintos uns dos outros. Além disso, a teoria musical aprendida e a coordenação motora obtida a partir do aprendizado de um certo instrumento podem ser utilizadas no aprendizado de outro.

2.5.2 Domínio Incremental

O cenário de domínio incremental é descrito como a situação em que a estrutura das tarefas é sempre a mesma, mas existe uma mudança do contexto ou da distribuição dos seus dados. Nesse caso, o modelo deve aprender uma série de tarefas, ou domínios, incrementalmente, mas, ao contrário do cenário de tarefa incremental, no momento da inferência, ele não sabe a qual tarefa o dado de entrada pertence. Em outras palavras, as tarefas têm as mesmas possíveis saídas, como classes, por exemplo, mas são apresentadas em diferentes contextos ou situações.

O uso de componentes específicos para cada tarefa não é possível neste cenário, pois o modelo não tem acesso ao identificador das tarefas e, consequentemente, não sabe a que tarefa ao dado de entrada pertence. Mas isso acaba não sendo necessário, uma vez que a estrutura das tarefas é sempre a mesma.

Um exemplo de aprendizado incremental de domínio seria uma aplicação de reconhecimento de faces, onde deve ser feito o reconhecimento de pessoas em ambientes internos e externos, com diferentes condições de iluminação e planos de fundo. Cada situação pode ser vista como um domínio diferente, e o modelo deve ser capaz de se adaptar a novos domínios incrementalmente sem esquecer o que aprendeu anteriormente. Nesse caso, cada tarefa teria dados referentes sempre as mesmas pessoas, variando apenas a situação na qual a imagem foi obtida. Neste cenário, o identificador da tarefa não está disponível no momento da inferência. Portanto, o modelo precisa ser capaz de reconhecer as faces independentemente do contexto em que elas aparecem, sem saber exatamente a qual domínio (ambiente interno, externo,

condições de iluminação, etc.) a imagem pertence. Isso torna o aprendizado contínuo neste cenário mais desafiador, pois o modelo deve ser capaz de generalizar com base na informação aprendida anteriormente, mesmo sem a presença de identificadores de tarefa explícitos.

2.5.3 Classe Incremental

O cenário de classe incremental é aquele em que o modelo deve aprender incrementalmente a distinguir entre um número crescente de classes. Neste cenário, um conjunto de tarefas é apresentado, onde cada tarefa contém diferentes classes e o modelo deve aprender a distinguir entre todas as classes. Esse é o cenário mais desafiador, pois a identificação da tarefa não está disponível, então o modelo deve identificar a classe dentre todas aquelas que já foram vistas até o momento. Em síntese, o modelo deve ser capaz de resolver cada tarefa individualmente, distinguindo entre as classes dentro de uma determinada tarefa, e identificar a qual tarefa o dado de entrada pertence.

Exemplo de aplicação no cenário de classe incremental na vida real seria um sistema de reconhecimento de atividades humanas, que precisaria aprender a identificar diferentes atividades, como caminhar, correr ou andar de bicicleta, em ordem incremental. À medida que novas atividades são adicionadas, o modelo utiliza o conhecimento prévio para melhorar o reconhecimento de atividades posteriores.

2.6 ABORDAGENS PARA IMPLEMENTAR APRENDIZADO CONTÍNUO

As abordagens em aprendizado contínuo são um conjunto de técnicas usadas para que as redes neurais aprendam e se adaptem de forma contínua a novas tarefas e dados, preservando o conhecimento adquirido anteriormente. Essas abordagens podem ser categorizadas em três grupos principais: as que são baseadas em regularização, arquitetura e *rehearsal* (LANGE et al., 2022).

Cada uma dessas abordagens possui suas próprias características e vantagens, proporcionando soluções distintas para enfrentar os desafios do aprendizado contínuo. A seleção e combinação adequadas dessas abordagens são fundamentais para alcançar um bom desempenho em diferentes aplicações e cenários de aprendizado contínuo.

2.6.1 Baseadas em Regularização

As abordagens baseadas em regularização são uma forma de mitigar o esquecimento catastrófico através da adição de termos de regularização à função de perda utilizada no treinamento do modelo. O intuito é impedir que, durante o aprendizado de novas tarefas, a atualização dos parâmetros do modelo afete negativamente o desempenho nas tarefas anteriores.

Os termos de regularização têm o objetivo de restringir mudanças nos pesos importantes para as informações já aprendidas, de forma a preservá-las, permitindo que o modelo mantenha as informações aprendidas anteriormente enquanto adquire novas informações. O desafio, nesse caso, é encontrar a melhor maneira de quantificar a importância de cada elemento da rede para o desempenho de cada tarefa.

2.6.2 Baseadas em Arquitetura

As abordagens baseadas em arquitetura visam alterar a arquitetura da rede para permitir a aprendizagem de novas tarefas sem afetar o desempenho nas tarefas anteriores. Isso é feito, por exemplo, adicionando módulos específicos para cada tarefa, de forma que quando uma nova tarefa é apresentada, os módulos correspondentes são treinados para resolvê-la. Tais módulos podem ser camadas adicionadas à rede ou até mesmo redes completas.

Dessa forma, para que seja possível a sua implementação, normalmente os métodos baseados nessa abordagem necessitam do identificador da tarefa, para que o modelo saiba qual módulo utilizar ao receber um dado de entrada. Nessa abordagem, o desafio é encontrar a maneira mais eficiente de aumentar o tamanho da rede e de compartilhar entre os módulos o conhecimento obtido com cada tarefa.

2.6.3 Baseadas em *Rehearsal*

Essas abordagens funcionam armazenando os dados de cada tarefa e usando-os em conjunto com os dados das novas tarefas durante o treinamento da rede neural. O desafio dos métodos baseados nessa abordagem é encontrar maneiras mais eficientes de selecionar os dados, de forma a escolher as amostras mais representativas, e de como reapresentar os dados armazenados à rede neural durante o treinamento de novas tarefas.

Uma variação da abordagem de *Rehearsal* é a *Pseudo-Rehearsal*, que se baseia em gerar

dados sintéticos que representem os dados que foram vistos pela rede durante o treinamento das tarefas passadas. Os dados gerados são chamados de "pseudo-exemplos" (*pseudo-samples*) e são usados durante o treinamento da rede nas novas tarefas. A ideia é que a rede neural possa manter o conhecimento relacionado as tarefas anteriores sem a necessidade de armazenar os seus dados. Com o surgimento e desenvolvimento das redes generativas, métodos utilizando essa abordagem para problemas de visão computacional começaram a ser desenvolvidos.

3 TRABALHOS RELACIONADOS

Este capítulo tem como objetivo fornecer uma visão abrangente do desenvolvimento histórico do campo de aprendizado contínuo. Ele é estruturado em duas seções: na primeira, são abordados os trabalhos pioneiros da área, que se baseiam em redes conexionistas ou em redes neurais rasas; na segunda, são abordados os estudos contemporâneos, já no contexto das redes neurais profundas, incluindo uma explicação do então método estado da arte, utilizado como base do método desenvolvido nesse trabalho.

3.1 INÍCIO

O final da década de 80 e a década de 90 foi um período de desânimo e desinteresse pela Inteligência Artificial (IA), marcando o segundo "Inverno da IA" (TOOSI et al., 2021). Nessa época, uma série de problemas e limitações levaram a falta de avanços significativos na resolução de problemas mais complexos e impediram que a IA atingisse expectativas que foram criadas em torno dela.

Como consequência, investimentos em pesquisa e desenvolvimento diminuíram drasticamente em muitos países, incluindo os Estados Unidos, que era o principal centro de pesquisa em IA na época e, com isso, muitos pesquisadores e cientistas acabaram deixando o campo da inteligência artificial em busca de outras áreas de pesquisa.

No entanto, o Inverno da IA chegou ao fim na segunda metade dos anos 1990, quando ocorreu um ressurgimento do interesse em IA devido ao surgimento de novas técnicas e algoritmos que permitiram avanços significativos no desenvolvimento das redes neurais. Entre eles, destaca-se o algoritmo de *backpropagation*, proposto em 1986 por *David Rumelhart, Geoffrey Hinton* e *Ronald Williams* (RUMELHART; HINTON; WILLIAMS, 1986). Esse é um algoritmo de treinamento supervisionado para redes neurais que permite ajustar os pesos das conexões entre as unidades da rede de forma a minimizar o erro entre as saídas da rede e as saídas desejadas. O algoritmo de *backpropagation* permitiu que as redes neurais fossem treinadas de forma mais eficiente, o que foi fundamental para o ressurgimento do interesse em IA e início do fim do Inverno da IA.

Enquanto alguns estudos apresentavam novas maneiras de aumentar as capacidades das redes de resolver problemas, outros identificavam limitações desses modelos. Foi o caso, por

exemplo, dos trabalhos de *McCloskey* e *Cohen* (MCCLOSKEY; COHEN, 1989) e o de *Ratcliff* (RATCLIFF, 1990), que demonstraram que o processo de aprendizagem de um novo conjunto de padrões, ou tarefas, pode causar a perda do conhecimento que as redes já haviam adquirido anteriormente. Eles mostraram que isso ocorre porque as redes neurais geralmente compartilham um conjunto de pesos para aprender diferentes tarefas, o que faz com que o aprendizado de novos dados interfira no conjunto de pesos relacionados a tarefas anteriores. Esse fenômeno ficou conhecido como esquecimento catastrófico, ou interferência catastrófica.

A partir da identificação desse efeito, diversos métodos foram propostos com o objetivo de reduzir a sua ocorrência. No próprio artigo em que descrevia o fenômeno, *Ratcliff* propôs um método, chamado de *Rehearsal Buffer Model* (RATCLIFF, 1990), para reduzir o esquecimento através do treinamento da rede com uma parte dos dados de padrões aprendidos no passado, em conjunto com os dados dos novos padrões a serem aprendidos.

Estudos posteriores propuseram alterações baseadas no método de *Ratcliff* (ROBINS, 1993), variando a maneira como os dados das tarefas anteriores eram selecionados e reapresentados as redes durante o aprendizado de novas tarefas. Os algoritmos baseados no método de *Ratcliff* ficaram conhecidos como métodos de *rehearsal* ou de *replay*.

Uma das limitações desses métodos é que eles exigem o armazenamento de todos ou de alguns dos dados de treinamentos anteriores para que possam ser usados em treinamentos futuros. Com isso, à medida que o tamanho do conjunto de dados aumenta, os custos computacionais e de armazenamento necessários para treinamento de um modelo se tornam cada vez maiores.

Além disso, o armazenamento explícito de dados de treinamento anteriores não é biologicamente plausível (MOE-HELGESEN; STRANDEN, 2005), uma vez que os humanos não precisam acessar novamente todas as informações aprendidas no passado para aprender novas informações. Portanto, espera-se que um método de aprendizado contínuo de redes neurais seja capaz de aprender continuamente com novos dados sem esquecer completamente os dados anteriores e sem a necessidade de acessá-los novamente, de forma semelhante à capacidade de aprendizado humano.

Por fim, o uso de técnicas de *rehearsal* também pode causar a ocorrência de *overfitting* e limitar a capacidade de generalização do modelo para novos dados (VERWIMP; LANGE; TUYTELAARS, 2021) caso apenas um mesmo subconjunto de dados anteriores seja utilizado durante treinamentos futuros.

Para evitar esses problemas, algumas outras abordagens foram propostas para resolver

ou reduzir o efeito de esquecimento, entre elas, a de *pseudorehearsal* ou *pseudoreplay*. Essa abordagem foi proposta inicialmente por (ROBINS, 1995) e se concentra em permitir que o aprendizado continuo possa ser feito em situações onde os dados de tarefas anteriores não estão mais acessíveis por meio da utilização de dados sintéticos ao invés dos dados reais.

A proposta inicial de *Robins* foi de gerar um conjunto de dados (*pseudosamples*) composto de vetores com permutações aleatórias com 50% de valores 0 e 50% de valores 1. Quando uma nova tarefa precisa ser aprendida, o conjunto de dados gerado é apresentado à rede, produzindo os vetores de saída correspondentes. As entradas aleatórias e suas saídas são então adicionadas ao conjunto de treinamento da rede e, após isso, a rede é treinada normalmente. A hipótese é que, da mesma forma que reapresentar os dados originais das tarefas passadas impede a rede de as esquecer, apresentar os *pseudosamples* que aproximam a função definida pelas tarefas aprendidas também preveniria o esquecimento catastrófico.

Além de eliminar o requisito de armazenamento dos dados, a abordagem de *pseudorehearsal* também é mais próxima da forma como ocorre biologicamente o aprendizado contínuo. Um modelo de aprendizado humano (MCCLELLAND; MCNAUGHTON; O'REILLY, 1995) mostra a relação entre o hipocampo e o neocórtex durante o processo de aprendizagem e sugere que os neurônios do neocórtex podem sofrer de esquecimento catastrófico. O modelo mostra que essa ocorrência é evitada através de um processo que reapresenta memórias armazenadas no hipocampo para reforçar tarefas que não foram realizadas recentemente.

Na sequência da descoberta do esquecimento catastrófico, surgiram diversas propostas de soluções para o problema. Além dos métodos baseados em *replay* e *pseudo-replay*, métodos baseados em outras abordagens também foram propostos. Entre eles, *French* foi um dos primeiros a propôr um método baseado em regularização, onde se busca reduzir a alteração de pesos da rede referentes a tarefas já aprendidas.

O método proposto por *French*, *Activation Sharpening* (FRENCH, 1992), se baseia na hipótese de que o esquecimento catastrófico é uma consequência direta da natureza distribuída de uma rede neural, onde quase todos os nós contribuem para o armazenamento de cada padrão.

O algoritmo funciona da seguinte forma: é feito um *forward pass* pela rede para identificar os k nós mais ativos. Em seguida, esses k nós são "afiados", ou seja, seus pesos são ajustados para torná-los ainda mais ativos e relevantes para a tarefa em questão. Após isso, a diferença na ativação desses nós afiados é usada como medida de erro, e propagada de volta pela rede usando *backpropagation*. Por fim, é feito um *forward* e *backward pass* baseado apenas na

função de perda de classificação.

Com isso, a distribuição do aprendizado é reduzida, ou seja, menos nós na rede têm seus pesos ajustados durante o treinamento. Isso ocorre porque, ao afiar apenas os k nós mais ativos, o ajuste nos pesos dos nós menos relevantes para a tarefa é reduzido, uma vez que a *backpropagation* muda muito pouco os pesos quando a ativação de um nó é próxima de zero.

O algoritmo *Growing Neural Gas* (GNG) (FRITZKE, 1994) foi um dos primeiros a propor o crescimento de uma rede com o objetivo de permitir o aprendizado a partir de distribuições de dados dinâmicas. O método é baseado nas *Neural Gas Networks*, redes inspiradas nos *self-organizing maps* (SOM), e propõe uma rede incremental capaz de aprender as relações topológicas importantes para uma dada distribuição de dados de entrada utilizando o algoritmo *competitive Hebbian learning* (CHL). A partir do aprendizado das relações importantes, o método é capaz de aprender continuamente adicionando unidades e conexões até que um determinado critério de performance seja alcançado. A principal proposição do método é a adição sequencial de novas unidades a uma rede inicialmente pequena, sendo essa adição determinada pelo algoritmo CHL.

3.2 PERÍODO RECENTE

Embora diversos métodos tenham sido propostos no passado, os algoritmos desenvolvidos não eram capazes de apresentar desempenho satisfatório em cenários reais (MOE-HELGESEN; STRANDEN, 2005). Dessa forma, a grande contribuição deixada pelos métodos pioneiros foi a identificação das abordagens que podem ser utilizadas para lidar com o problema do esquecimento catastrófico (REBUFFI et al., 2017).

Após um período de pouco interesse, o problema do esquecimento voltou a receber atenção e novos métodos começaram a ser propostos após a *deep learning renaissance* (GOODFELLOW et al., 2013). Entre eles, o algoritmo *Elastic Weight Consolidation* (EWC) (KIRKPATRICK et al., 2017), que se baseia na abordagem de regularização, de forma que as conexões da rede responsáveis por tarefas anteriores sejam preservadas. Para isso, os autores propõem o uso da matriz de *Fisher* para estimar a importância de cada parâmetro do modelo em relação às tarefas passadas. Eles introduzem um componente na função de perda que utiliza essa estimativa para ponderar a penalidade aplicada à variação dos parâmetros atuais da rede em relação aos parâmetros encontrados durante o treinamento das tarefas anteriores. Ou seja, ao treinar uma nova tarefa, a variação dos parâmetros é penalizada de acordo com a sua

importância para as tarefas passadas, que é determinada pela matriz de Fisher.

A matriz de Fisher quantifica a informação que um conjunto de variáveis aleatória (os dados) fornece sobre os parâmetros desconhecidos de uma distribuição de probabilidade (aproximada pelo modelo). Ela faz isso medindo a curvatura da função de log-verossimilhança em relação aos parâmetros do modelo. A função de log-verossimilhança representa a probabilidade de observar os dados, dado um modelo e um conjunto de parâmetros. Uma maior curvatura na função log-verossimilhança indica que os dados fornecem mais informações sobre os parâmetros, permitindo fazer estimativas mais precisas desses parâmetros.

O algoritmo EWC utiliza a matriz de Fisher para aproximar a curvatura da função log-verossimilhança em relação aos parâmetros do modelo. Isso é feito calculando a matriz para os parâmetros do modelo após o treinamento em uma tarefa, capturando a sensibilidade das probabilidades previstas em relação aos parâmetros. Dessa forma, a matriz pode ser usada para estimar a importância de cada parâmetro para essa tarefa.

Considerando a abordagem de *rehearsal*, o algoritmo iCaRL (*Incremental Classifier and Representation Learning*) utiliza o conceito de amostras "exemplares", que são um subconjunto dos dados de treinamento de tarefas anteriores. Os exemplares são as amostras que mais se aproximam da representação média das classes, e são usados para manter a representação das classes antigas enquanto novas classes são aprendidas. A quantidade de exemplares é definida de acordo com a quantidade de memória disponível para armazenamento.

Ao aprender uma nova tarefa, o iCaRL atualiza a rede usando tanto os novos dados quanto os exemplares das tarefas anteriores. Além da função de perda *softmax* tradicional usada para treinar a rede para classificar corretamente os dados, o iCaRL utiliza também um componente para minimizar a distância entre a representação dos exemplares e o centróide de sua classe correspondente.

O iCaRL usa a estratégia de classificação “média mais próxima dos exemplares” onde, para estimar a classe de uma dada amostra de entrada, o algoritmo primeiro calcula um vetor protótipo para todas as classes já observadas a partir da média da representação de seus protótipos e, em seguida, determina o vetor de características da amostra que precisa ser classificada. Com base nisso, o iCaRL atribui à amostra o rótulo da classe cujo vetor protótipo apresenta a maior similaridade com o seu vetor de características.

Já no contexto de métodos baseados em arquitetura, as Redes Neurais Progressivas (*Progressive Neural Networks* - PNN) mantêm uma rede (chamada de coluna) para cada tarefa e, ao aprender uma nova tarefa, adiciona uma nova rede à estrutura sem alterar as redes

existentes.

O treinamento de uma PNN começa com uma única rede, ou coluna, e é treinada até a convergência com os dados da primeira tarefa. Ao receber uma nova tarefa para treinamento, os parâmetros da rede anterior são "congelados" e uma nova coluna é adicionada com uma inicialização aleatória. Esta nova coluna é treinada não apenas com os dados da tarefa atual, mas também com as saídas das camadas da coluna anterior por meio de conexões laterais. Cada nova tarefa é tratada por uma nova coluna, e essas colunas são conectadas lateralmente para permitir a transferência de conhecimento.

As conexões laterais permitem que a nova coluna de tarefas acesse e utilize características aprendidas pelas colunas anteriores. Isso permite que a rede reutilize, modifique ou ignore as características aprendidas anteriormente, conforme necessário para a nova tarefa. Como as conexões laterais são apenas da coluna atual para as colunas anteriores, e os parâmetros das colunas anteriores são mantidos congelados durante o treinamento da nova coluna, não há interferência entre as tarefas.

3.2.1 Estado da Arte

3.2.2 IRCL

No artigo *Learning Invariant Representation for Continual Learning* (SOKAR; MOCANU; PECHENIZKIY, 2021), foi proposto um método baseado em *pseudo-rehearsal* chamado *Invariant Representation for Continual Learning* (IRCL), o qual é a base do método proposto nessa dissertação. O foco do trabalho é o cenário de aprendizado de classe incremental, onde o modelo não tem acesso a identidade da tarefa. Os autores propõem a utilização da representação dos dados dividida em duas partes: uma representação invariante e uma representação discriminante. A representação invariante captura as características comuns entre as classes, enquanto a representação discriminante se concentra nas características específicas de cada classe.

A razão para usar uma representação desvinculada é que a representação invariante é menos propensa ao esquecimento, além de capturar características que podem ser úteis durante o aprendizado de novas tarefas (EBRAHIMI et al., 2020).

O método IRCL consiste em utilizar uma arquitetura unificada, ilustrada na Figura 15, para classificação e geração de dados. A parte de classificação da arquitetura é composta por um

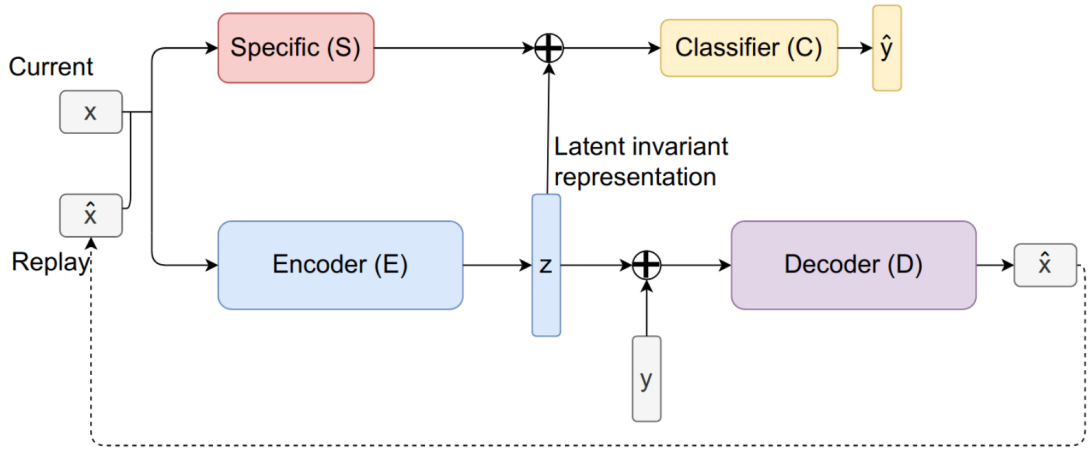


Figura 15 – Arquitetura proposta no artigo *Learning Invariant Representation for Continual Learning*

módulo extrator de características específicas (S), que gera uma representação discriminativa dos dados de entrada, e um módulo de classificação (C) que aprende a classificar os dados da sequência de tarefas com base nas representações discriminante e invariante.

A Equação 3.1 representa o objetivo do treinamento dos módulos de classificação da arquitetura: a minimização da função de perda $\mathcal{L}_C$, em função dos parâmetros do extrator de características específicas (θ_S) e do módulo de classificação (θ_C). $\mathcal{L}_C$ é a função de perda que mede o quão bem o modelo está classificando os dados, z é a representação invariante, D^t é o conjunto de dados de treinamento da tarefa atual t e $M^{1:t-1}$ é o conjunto de dados gerados referentes as tarefas anteriores.

$$\min_{\theta_S,\theta_C} \mathcal{L}_C(\theta_S, \theta_C; z, D^t \cup \mathcal{M}^{1:t-1}) \tag{3.1}$$

A parte de geração de dados da rede, um *Autoencoder* Variacional Condicional (cVAE), é composta por um codificador (*encoder* - E) que mapeia uma entrada x para uma representação latente $z \sim p(z|x)$, e um decodificador (*decoder* - D) que mapeia a saída do codificador combinada ao rótulo da classe de entrada y de volta para o espaço de entrada $\hat{x} \sim p(x|z, y)$. O espaço latente é regularizado de forma a seguir uma distribuição normal, permitindo a extração de amostras dele e a geração de novos dados usando o decodificador, condicionando-o às classes das tarefas anteriores.

A Equação 3.2 representa a função de perda otimizada no treinamento do cVAE. Ela é composta por dois termos principais, onde o primeiro, $||x - \hat{x}||^2$, é a diferença quadrática entre os dados de entrada x e os dados reconstruídos $\hat{x}$. Esse termo penaliza o modelo com base no erro de reconstrução dos dados de entrada, incentivando o cVAE a aprender uma

representação latente que possa reconstruir efetivamente os dados de entrada.

$$\mathcal{L}_{cVAE} = ||x - \hat{x}||^2 + KL(q(z|x)||p(z)) \tag{3.2}$$

O segundo termo, $KL(q(z|x)||p(z))$, é a divergência de *Kullback-Leibler* (KL), que penaliza o modelo de acordo com a divergência entre a representação latente gerada e a distribuição normal, forçando o cVAE a aprender uma representação latente que siga essa distribuição.

A combinação desses dois termos na função objetivo permite que o cVAE aprenda efetivamente representações latentes que podem ser usadas para reconstruir os dados de entrada, além de permitir que novas representações sejam geradas através da amostragem de um vetor aleatório a partir de uma distribuição normal.

Durante o treinamento, o cVAE aprende a produzir a representação invariante que capta as características comuns a todas as classes e, em seguida, aprende a mapear essa representação de volta ao espaço de entrada. Enquanto isso, o extrator de características específicas (S) aprende a gerar uma representação específica de cada classe que capta suas características exclusivas. A representação específica é então combinada com a representação invariante para servir como entrada para o módulo classificador (C).

Antes de iniciar o treinamento de uma nova tarefa, o decodificador atua como o componente de memória da arquitetura, gerando dados referentes às tarefas anteriores que são então combinados ao conjunto de treinamento da tarefa atual. Dessa forma, durante o treinamento da nova tarefa, os dados gerados permitem à rede "lembrar" como eram os dados passados, evitando assim o esquecimento e mantendo a rede capaz de classificar os dados de classes de tarefas anteriores.

Em síntese, o IRCL pode ser representado pelo pseudo código a seguir. Note que, por questão de objetividade, foram abstraídos alguns detalhes, como a definição dos parâmetros, $anotacoes$ e $dimensao\_features$ na chamada da funçao $GERAR\_IMGS\_ANTERIORES$, assim como a definição do tamanho do vetor na chamada da função $ONEHOT\_ENCODING$ e o número de épocas de treinamento.

Algorithm 1 IRCL - Invariant Representation for Continual Learning

```
function APRENDIZADO_CONTINUO(imgs_atuais, labels_atuais)
    if primeira_tarefa then
        modelo ← NOVOMODELO()
        modelo ← APRENDER_TAREFA(modelo, imgs_atuais, labels_atuais)
    else
        imgs_anteriores, labels_anteriores ← GERAR_IMGS_ANTERIORES(modelo)
        imgs ← CONCATENAR(imgs_atuais, imgs_anteriores)
        labels ← CONCATENAR(labels_atuais, labels_anteriores)
        modelo ← APRENDER_TAREFA(modelo, imgs, labels)
    end if
    return modelo
end function

function APRENDER_TAREFA(modelo, imgs, labels)
    for epoca in epocas do
        labels_onehot ← ONEHOT_ENCODING(labels)
        features_encoder ← modelo.ENCODER(imgs)
        entrada_decoder ← CONCATENAR(features_encoder, labels_onehot)
        imgs_geradas ← modelo.DECODER(entrada_decoder)
        erro_geracao ← FUNCAO_DE_PERDA_GERACAO(imgs, imgs_geradas)
        modelo.ATUALIZAR_PESOS(erro_geracao)

        features_encoder ← modelo.ENCODER(imgs)
        features_especifico ← modelo.ESPECIFICO(imgs)
        features ← CONCATENAR(features_especifico, features_encoder)
        predicao ← modelo.CLASSIFICADOR(features)
        erro_predicao ← FUNCAO_DE_PERDA_CLASSIFICACAO(labels, predicao)
        modelo.ATUALIZAR_PESOS(erro_predicao)
    end for
    return modelo
end function

function GERAR_IMGS_ANTERIORES(modelo)
    features_aleatorias ← GERARVETORESALEATORIOS(dimensao_features)
    anotacoes_onehot ← ONEHOT_ENCODING(anotacoes)
    entrada_decoder ← CONCATENAR(features_aleatorias, anotacoes_onehot)
    imagens_geradas ← modelo.DECODER(entrada_decoder)
    return imagens_geradas, anotacoes
end function
```

4 MÉTODO PROPOSTO

O método proposto no trabalho apresentado nessa dissertação se baseia no método IRCL e parte da premissa de que, em uma abordagem de pseudo-replay, o modelo generativo é o principal limitador para o desempenho da aprendizagem contínua (SHIN et al., 2017). A intuição por trás disso é que, ao considerar um gerador ideal capaz de criar imagens artificiais idênticas às reais, isso seria equivalente ao treinamento do modelo utilizando todos os dados reais como uma única tarefa, ou seja, em um cenário de aprendizagem não contínua.

Este trabalho visa aprimorar o desempenho da aprendizagem contínua apresentado pelo IRCL, através da melhoria da similaridade entre as imagens geradas e as reais, ao mesmo tempo em que evita o problema do esquecimento catastrófico no modelo generativo. Para isso, propomos alterações no método de treinamento, no modelo gerador utilizado e no tipo de camada utilizada na implementação da arquitetura proposta no artigo do IRCL.

4.1 HIPÓTESES

4.1.1 Treinamento desacoplado

No método IRCL, todos os módulos da rede, incluindo o autoencoder, o módulo classificador e módulo específico, são treinados em conjunto. Essa abordagem de treinamento, que nós chamamos de acoplada, exige a escolha de um conjunto de hiperparâmetros que seja capaz de proporcionar um bom desempenho geral para todos os módulos. No entanto, esse conjunto de hiperparâmetros pode não ser ideal para obter os melhores resultados individuais em cada um dos módulos. Em outras palavras, ao buscar um bom desempenho global, o desempenho individual dos componentes da rede pode ser comprometido.

Uma vez que não existe retropropagação entre os módulos de geração de imagens e os demais módulos da arquitetura, elaboramos a hipótese de que desacoplar o treinamento do cVAE dos demais módulos permitiria utilizar um conjunto ótimo, ou quase ótimo, de hiperparâmetros para o treinamento de cada um deles. Ao fazer isso a expectativa é de que, ao final do treinamento, tenhamos modelos com melhores capacidades de extração de características e de geração de imagens, resultando em um melhor desempenho final na classificação e na retenção do conhecimento passado.

4.1.2 cVAE-GAN como Rede Generativa

O Autoencoder Variacional Condicional (cVAE) utilizado no IRCL possui propriedades desejáveis para o contexto de aprendizagem contínua baseada em *pseudo-replay*. Primeiramente, por ser variacional, seu espaço latente segue uma distribuição pré-definida a partir da qual é possível gerar novas amostras de dados. Em segundo lugar, por ser condicional, o modelo pode ser usado para gerar dados de todas as classes aprendidas até então. Além disso, essa característica induz o codificador a produzir uma representação invariante, uma vez que a informação referente a classe é fornecida como entrada ao decodificador e não precisa ser codificada. Por último, a representação invariante é menos propensa a esquecimento por não conter informações específicas de cada classe e é especialmente útil no aprendizado de novas tarefas (EBRAHIMI et al., 2020).

Apesar das propriedades benéficas, os VAEs têm a tendência de produzir imagens com baixa nitidez (LARSEN et al., 2016), o que é indesejado devido à dependência que a abordagem de *pseudo-replay* tem em relação à qualidade das amostras geradas. Portanto, para alcançar um melhor desempenho na aprendizagem contínua, é desejável ter um modelo gerativo capaz de produzir imagens de maior qualidade e que mantenha as propriedades positivas dos VAEs. Para isso, elaboramos a hipótese de que o treinamento desacoplado do modelo generativo permitiria utilizar um modelo generativo mais robusto, como um cVAE-GAN, capaz de gerar imagens mais nítidas, ao mesmo tempo em que é condicional e produz representações invariantes. A partir disso, propomos a arquitetura apresentada na Figura 16.

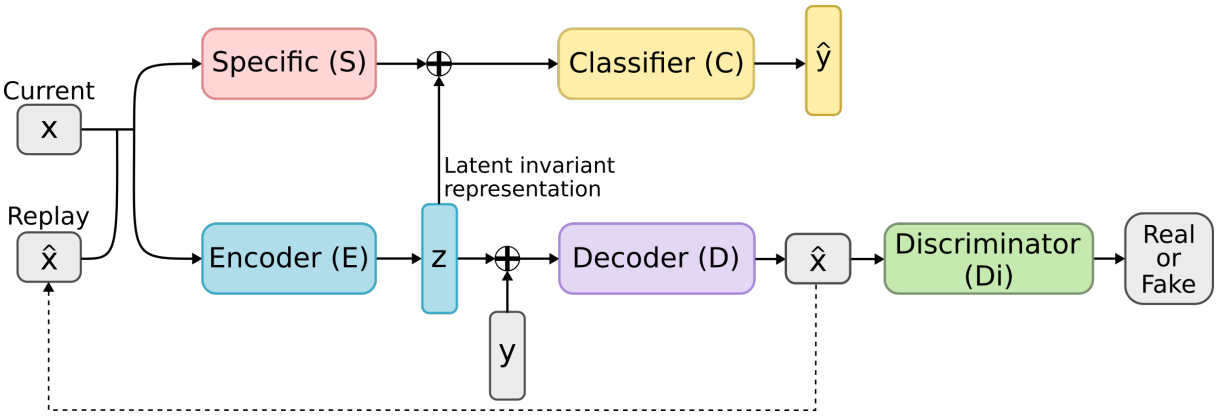


Figura 16 – Arquitetura proposta

Em síntese, o treinamento desacoplado com uso de um cVAE-GAN é representado na função $APRENDER\_TAREFA$ do pseudo código a seguir. Note que, assim como no pseudo código anterior, alguns detalhes foram abstraídos por questão de objetividade. Aqui,

além dos detalhes citados anteriormente, foi omitida também a definição das anotações usadas para computar a $FUNCAO\_PERDA\_DISCRIMINACAO$.

Algorithm 2 Treinamento desacoplado + cVAE-GAN

function APRENDIZADOCONTINUO($imgs\_atuais$, $labels\_atuais$)
  if primeira_tarefa **then**
    $modelo \leftarrow$ NOVOMODELO()
    $modelo \leftarrow$ APRENDER_TAREFA($modelo, imgs\_atuais, labels\_atuais$)
  else
    $imgs\_anteriores, labels\_anteriores \leftarrow$ GERAR_IMGS_ANTERIORES($modelo$)
    $imgs \leftarrow$ CONCATENAR($imgs\_atuais, imgs\_anteriores$)
    $labels \leftarrow$ CONCATENAR($labels\_atuais, labels\_anteriores$)
    $modelo \leftarrow$ APRENDER_TAREFA($modelo, imgs, labels$)
  end if
  return $modelo$
end function

function APRENDER_TAREFA($modelo$, $imgs$, $labels$)
  for $epoca$ in $epocas$ **do**
    $labels\_onehot \leftarrow$ ONEHOT_ENCODING($labels$)
    $features\_encoder \leftarrow modelo$.ENCODER($imgs$)
    $entrada\_decoder \leftarrow$ CONCATENAR($features\_encoder, labels\_onehot$)
    $imgs\_geradas \leftarrow modelo$.DECODER($entrada\_decoder$)
    $discriminacao \leftarrow modelo$.DISCRIMINADOR($imgs, imgs\_geradas$)
    $erro\_geracao \leftarrow$ FUNCAO_DE_PERDA_GERACAO($imgs, imgs\_geradas$)
    $erro\_discriminacao \leftarrow$ FUNCAO_PERDA_DISCRIMINACAO($discriminacao$)
    $modelo$.ATUALIZAR_PESOS($erro\_geracao, erro\_discriminacao$)
  end for

  for $epoca$ in $epocas$ **do**
    $features\_encoder \leftarrow modelo$.ENCODER($imgs$)
    $features\_especifico \leftarrow modelo$.ESPECIFICO($imgs$)
    $features \leftarrow$ CONCATENAR($features\_especifico, features\_encoder$)
    $predicao \leftarrow modelo$.CLASSIFICADOR($features$)
    $erro\_predicao \leftarrow$ FUNCAO_DE_PERDA_CLASSIFICACAO($labels, predicao$)
    $modelo$.ATUALIZAR_PESOS($erro\_predicao$)
  end for
  return $modelo$
end function

function GERAR_IMGS_ANTERIORES($modelo$)
  $features\_aleatorias \leftarrow$ GERARVETORESALEATORIOS($dimensao\_features$)
  $anotacoes\_onehot \leftarrow$ ONEHOT_ENCODING($anotacoes$)
  $entrada\_decoder \leftarrow$ CONCATENAR($features\_aleatorias, anotacoes\_onehot$)
  $imagens\_geradas \leftarrow modelo$.DECODER($entrada\_decoder$)
  return $imagens\_geradas$, $anotacoes$
end function

4.1.3 Uso de Camadas Convolucionais

A implementação do oficial do IRCL, disponibilizada pelos autores, utiliza camadas lineares em todos os módulos de sua arquitetura. No entanto, como discutido na subseção 2.2.2, camadas convolucionais possuem uma melhor capacidade de lidar com dados que possuem estruturas espaciais, como imagens.

Com base nisso, propomos implementar os módulos da arquitetura utilizando camadas convolucionais, visando aprimorar a extração de características e a geração de imagens. Com essa abordagem, esperamos obter melhorias no desempenho da classificação e na conservação do conhecimento referente as tarefas anteriores.

4.2 CONJUNTOS DE DADOS

O *MNIST* e *Fashion-MNIST* são conjuntos de dados (*datasets*) amplamente utilizados para avaliar e comparar o desempenho de algoritmos e modelos de aprendizado de máquina em tarefas de visão computacional, como classificação de imagens.

O *MNIST* é composto por 70.000 imagens de dígitos manuscritos entre 0 e 9 dividido em 60.000 imagens de treinamento e 10.000 imagens de teste, com uma resolução de 28x28 *pixels* em escala de cinza. Exemplos de imagens[1] presentes no MNIST são apresentadas na Figura 17.

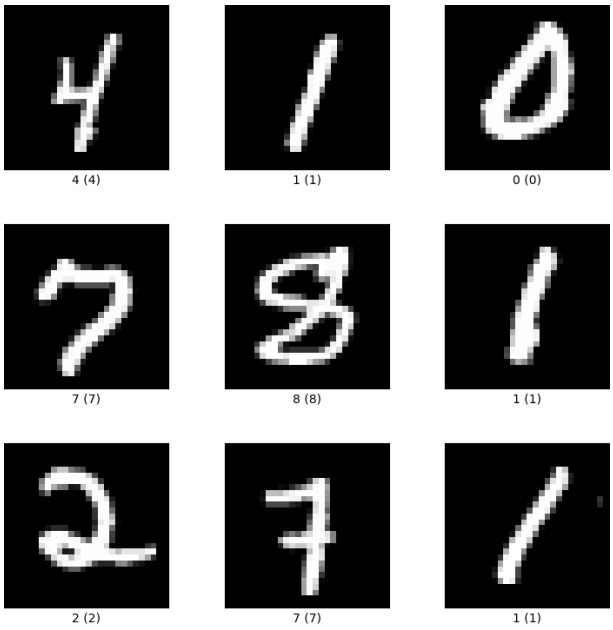


Figura 17 – Exemplos de imagens do MNIST

[1] *TensorFlow - MNIST*

O *Fashion-MNIST* foi criado como uma alternativa ao *MNIST*, com o objetivo de fornecer um desafio mais complexo para os algoritmos. Assim como o *MNIST*, ele consiste em 70.000 imagens em escala de cinza de 28x28 *pixels*, porém, ao invés de dígitos manuscritos, ele representa 10 classes de artigos de vestuário, como camisetas, calças, casacos e vestidos. Assim como o *MNIST*, o *Fashion-MNIST* também é dividido em um conjunto de treinamento com 60.000 imagens e um conjunto de teste com 10.000 imagens. No entanto, o *Fashion-MNIST* apresenta uma maior diversidade de padrões, formas e texturas em comparação ao *MNIST*. A Figura 18 apresenta algumas imagens[2] presentes no Fashion-MNIST.

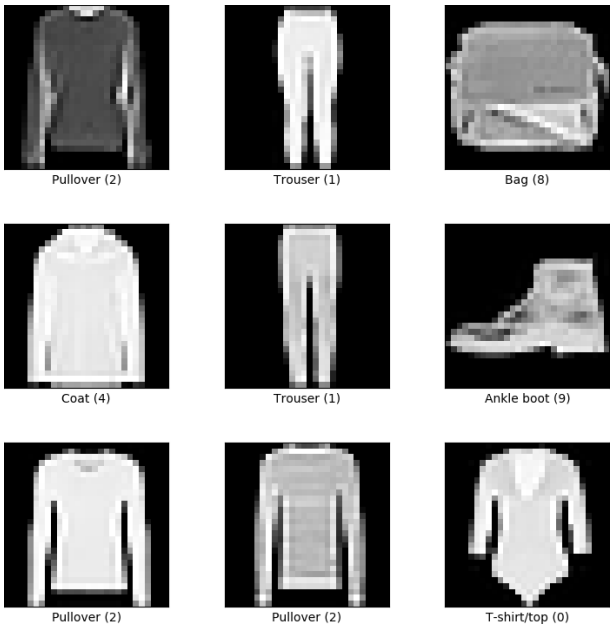


Figura 18 – Exemplos de imagens do Fashion-MNIST

O *Split MNIST* e o *Split Fashion-MNIST* são variações do *MNIST* e *Fashion-MNIST* utilizadas para avaliar métodos de aprendizado contínuo. A diferença está no fato de que os *datasets* são divididos em tarefas, onde cada uma delas contém um conjunto de classes dos dados originais. Neste trabalho, o *Split MNIST* e *Split Fashion-MNIST* foram definidos contendo 5 tarefas com 2 classes cada. A Figura 19 ilustra como foi feita a divisão do MNIST em tarefas.

Figura 19 – Divisão de classes entre tarefas para o Split MNIST

[2] *TensorFlow - Fashion-MNIST*

4.3 MÉTRICAS DE AVALIAÇÃO

4.3.1 Média da Acurácia de Classificação

A Média da Acurácia de Classificação (Acc_{avg}) é uma medida que quantifica o desempenho de um modelo na classificação de dados de tarefas aprendidas sequencialmente. Ela é uma média simples das acurácias em cada tarefa individual, calculadas após o treinamento de todas as tarefas. A Equação 4.1 representa a Acc_{avg}, onde N é a quantidade de tarefas e $A_{N,i}$ é a acurácia na $i-\acute{e}sima$ tarefa após o treinamento de todas as N tarefas.

$$Acc_{avg} = \frac{1}{N}\sum_{i=1}^{N} A_{N,i} \tag{4.1}$$

Modelos com alta ACC são considerados melhores no aprendizado contínuo, pois conseguem aprender novas tarefas e manter o desempenho nas tarefas anteriores.

4.3.2 Média do *Backward Transfer*

A métrica "*Backward Transfer*" (BWT) é usada para avaliar o impacto do aprendizado de novas tarefas no desempenho de um modelo em tarefas anteriores. Ela quantifica a capacidade do modelo de manter ou até melhorar seu desempenho nas tarefas anteriores, enquanto aprende novas tarefas.

Valores de BWT positivos indicam que o aprendizado de novas tarefas ajudou a melhorar o desempenho nas tarefas anteriores, um efeito conhecido como transferência positiva. Valores negativos indicam que o aprendizado de novas tarefas prejudicou o desempenho nas tarefas anteriores, o que sugere esquecimento catastrófico. Um valor de BWT próximo a zero indica que o desempenho nas tarefas anteriores não foi significativamente afetado pelo aprendizado de novas tarefas. Modelos com alto valor de BWT são considerados melhores no aprendizado contínuo, pois conseguem lidar com novas tarefas sem afetar negativamente o desempenho nas tarefas anteriores. A BWT é representada na Equação 4.2, onde $A_{i,i}$ é a acurácia na $i-\acute{e}sima$ tarefa logo após o seu treinamento, e $A_{N,i}$ é a acurácia na $i-\acute{e}sima$ tarefa após o treinamento de todas as N tarefas.

$$BWT_{avg} = \frac{1}{N-1}\sum_{i=1}^{N-1}(A_{N,i} - A_{i,i}) \tag{4.2}$$

4.3.3 *Medida do Índice de Similaridade Estrutural (SSIM)*

A métrica *Structural Similarity Index Measure* (SSIM) é uma medida de qualidade de imagem que avalia a similaridade estrutural entre duas imagens. Diferente das métricas baseadas em erro, como o erro quadrático médio (MSE) ou o erro absoluto médio (MAE), que comparam a diferença de intensidade de pixel entre as imagens, o SSIM considera a percepção humana e leva em conta a luminância, o contraste e a estrutura das imagens.

O SSIM é calculado em uma janela deslizante que compara regiões das duas imagens, gerando um valor para cada região comparada. O valor do SSIM varia entre -1 e 1, onde 1 indica que as imagens são idênticas e valores menores indicam uma menor similaridade estrutural. Ao calcular a média dos valores de SSIM de todas as regiões comparadas, obtém-se um único valor que indica a similaridade estrutural geral entre as duas imagens. Esta métrica é amplamente utilizada para avaliar a qualidade de imagens comprimidas, restauradas ou geradas por algoritmos de aprendizado de máquina.

O SSIM é representado pela Equação 4.3 a seguir:

$$SSIM(x, y) = l(x, y)^{\alpha} \cdot c(x, y)^{\beta} \cdot s(x, y)^{\gamma} \tag{4.3}$$

onde x e y são as imagens comparadas ou regiões delas, l, c e s são funções de comparação para luminância, contraste e estrutura, respectivamente, e α, β e γ são os pesos para cada função de comparação.

A função de luminância l compara a média das intensidades dos pixels entre duas imagens, buscando avaliar se as duas imagens têm brilho similar. Ela é definida por:

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

onde μ_x e μ_y são as médias de intensidade dos pixels das imagens x e y, respectivamente, e C_1 é uma constante pequena para evitar divisão por zero.

A função de contraste c mede a compatibilidade entre os desvios padrão das intensidades dos pixels das duas imagens, refletindo se as duas imagens têm um grau similar de contraste. Ela é expressa por:

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

onde σ_x e σ_y são os desvios padrão de x e y, respectivamente, e C_2 é uma constante pequena para evitar divisão por zero.

A função de estrutura s compara a covariância entre as duas imagens em relação aos seus desvios padrão, buscando avaliar se as duas imagens compartilham padrões de variação espacial (ou estrutura). Esta função é dada por:

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$$

onde σ_{xy} é a covariância entre x e y, e C_3 é uma constante pequena. Normalmente $C_3 = C_2/2$ para simplificar.

5 EXPERIMENTOS E RESULTADOS

Conforme discutido no Capítulo 4, o método apresentado nessa dissertação se baseia no trabalho do artigo *Learning Invariant Representation for Continual Learning* (IRCL). Portanto, nós comparamos diretamente nosso método com os resultados do IRCL. Por conta disso, ao desenvolver nosso método, também implementamos a arquitetura e a abordagem de treinamento originais do IRCL para reproduzir seus resultados e estabelecer uma referência de desempenho. Para o comparativo, realizamos experimentos para verificar o efeito da aplicação das hipóteses elaboradas.

Na implementação oficial[1] do artigo disponibilizada pelos autores, os módulos da arquitetura do IRCL possuem as seguintes características: o módulo específico (S) tem uma camada oculta de 20 neurônios, o módulo classificador (C) tem uma camada com 40 neurônios, o codificador (E) e decodificador (D) possuem uma camada oculta de 300 neurônios, com uma representação latente de tamanho 32. Todos os módulos utilizam camadas lineares e função de ativação ReLU.

Para reproduzir os resultados, treinamos a arquitetura utilizando os mesmos hiperparâmetros reportados no artigo, ou seja, usando Adam como otimizador, treinando por 5 e 10 épocas para o Split MNIST e Split FashionMNIST, respectivamente, utilizando um tamanho de *batch* de 128 e taxas de aprendizado de 2×10^{-4} para os módulos específico e classificador[2], e 10^{-2} para o cVAE. Ao treinar uma nova tarefa, foram geradas 5000 pseudo-amostras de cada uma das classes anteriores.

Em todos os experimentos, utilizamos a mesma semente para os geradores de números aleatórios das bibliotecas durante a implementação. Isso garante que as variações observadas nos resultados não são resultado de fatores vinculados à aleatoriedade, como por exemplo, a inicialização dos pesos das redes.

---

[1] *Implementation for the paper "Learning Invariant Representation for Continual Learning" in PyTorch.*

[2] Embora no artigo seja reportado 10^{-2}, no código, na verdade, é utilizado 2×10^{-4}. Usar 10^{-2} não resulta nas métricas reportadas pelos autores.

5.1 RESULTADOS

5.1.1 Treinamento desacoplado

Neste experimento, preservamos a arquitetura original proposta pelo IRCL, alterando apenas a sua forma de treinamento. O objetivo é verificar se desacoplar o treinamento do cVAE dos demais módulos da arquitetura permite utilizar hiperparâmetros que resultem em modelos com melhores capacidades de extração de características, geração de imagens e, consequentemente, melhor desempenho na classificação de imagens e retenção do conhecimento passado.

Para isso, decidimos treinar primeiro o cVAE, pois a representação intermediária gerada pelo seu codificador (E) é utilizada pelo módulo de classificação (C). Na sequência, os módulos específicos (S) e de classificação (C) foram treinados em conjunto. Os hiperparâmetros usados nesse experimentos são apresentados na Tabela 1.

Tabela 1 – Hiperparâmetros utilizados

Conjunto de Dados	**Módulo**	**Épocas**	**Taxa de Aprendizado**
Split MNIST	Específico	5	2×10^{-4}
	Classificação		
	cVAE	40	4×10^{-3}
Split FashionMNIST	Específico	5	2×10^{-4}
	Classificação		
	cVAE	25	10^{-2}

A Tabela 2 apresenta os resultados obtidos nos experimentos para cada método e conjunto de dados. É possível observar que o método de treinamento desacoplado proporcionou nas métricas quando comparado ao método original.

Tabela 2 – Comparação das Médias do SSIM, do BWT e da Acurácia

Método de Treino	**Split MNIST**			**Split FashionMNIST**		
	SSIM	BWT (%)	Acurácia (%)	SSIM	BWT (%)	Acurácia (%)
Original	0,599	**-11,314**	87,143	0,517	-23,494	76,070
Desacoplado	**0,643**	-11,680	**87,227**	**0,519**	**-17,887**	**77,940**

Em relação ao SSIM, o treinamento desacoplado resultou em uma melhora em ambos os conjuntos de dados, indicando uma melhoria na geração de imagens. Isso implica que, ao treinar o cVAE de forma isolada dos outros módulos, foi possível obter uma representação latente que codifica melhor as características das imagens.

Já em relação ao BWT, notou-se um comportamento distinto entre os conjuntos de dados. Enquanto no Split FashionMNIST o método desacoplado apresentou uma grande redução do fenômeno de esquecimento, indicado por valores de BWT menos negativos, o mesmo não foi observado no Split MNIST onde, na verdade, houve uma pequena piora, com valores mais negativos.

Apesar dos resultados mistos do BWT, o treinamento desacoplado levou a uma melhora geral de acurácia em ambos os casos, embora pequena para o conjunto de dados Split MNIST e mais notável no Split Fashion-MNIST. Esse incremento indica que desacoplar o treinamento estimula o módulo classificador a desenvolver uma melhor generalização para novas amostras, talvez devido a representações mais robustas fornecida pelo cVAE.

Em síntese, os resultados obtidos mostram que desacoplar o treinamento dos módulos da arquitetura leva a uma melhora na acurácia de classificação das imagens.

5.1.2 cVAE-GAN como Rede Generativa

Neste experimento, utilizamos o treinamento desacoplado para treinar um cVAE-GAN no lugar no cVAE usado originalmente na arquitetura do IRCL. O objetivo é verificar se o desacoplamento viabiliza o treinamento de modelos generativos mais robustos, com melhor capacidade de geração de imagens, levando a uma melhora no desempenho geral da arquitetura.

Para implementar o cVAE-GAN, adicionamos à arquitetura uma rede discriminadora após o decodificador, composta por quatro camadas convolucionais e uma camada linear de saída. Utilizamos *batch normalization* nas três camadas escondidas, usamos a ReLU como função de ativação das camadas convolucionais e Sigmoid na camada de saída. Os hiperparâmetros utilizados são apresentados na Tabela 3.

A Tabela 4 apresenta os resultados obtidos, onde é possível constatar que a adição do cVAE-GAN influenciou de maneira distinta as métricas entre os dois conjuntos de dados testados.

Com relação ao SSIM, o cVAE-GAN levou à uma redução na qualidade das imagens geradas para o conjunto Split MNIST, enquanto que houve uma melhora na qualidade das imagens geradas para o Split Fashion-MNIST. Para o BWT, a adição do cVAE-GAN teve um efeito positivo no Split MNIST, mas, no FashionMNIST, houve um aumento na taxa de esquecimento. Por fim, em relação a Acurácia, observamos uma melhora no Split MNIST, enquanto no Split FashionMNIST houve uma piora.

Tabela 3 – Hiperparâmetros utilizados

Conjunto de Dados	Módulo	Épocas	Taxa de Aprendizado
Split MNIST	Específico	5	2×10^{-4}
	Classificador		
	Codificador	100	2×10^{-4}
	Decodificador		
	Discriminador	100	10^{-5}
Split FashionMNIST	Específico	15	10^{-4}
	Classificador		
	Codificador	125	10^{-3}
	Decodificador		
	Discriminador	70	10^{-4}

Tabela 4 – Comparação das Médias do SSIM, do BWT e da Acurácia

Método de Treino	Split MNIST			Split FashionMNIST		
	SSIM	BWT (%)	Acurácia (%)	SSIM	BWT (%)	Acurácia (%)
Original	0,599	-11,314	87,143	0,517	-23,494	76,070
Desacoplado	**0,643**	-11,680	87,227	0,519	**-17,887**	**77,940**
Desacoplado cVAE-GAN	0,618	**-9,554**	**88,761**	**0,547**	-21,964	77,160

Em suma, a inclusão do cVAE-GAN na estrutura de treinamento desacoplado ofereceu vantagens para o conjunto de dados Split MNIST, mas o resultado positivo não se manteve para o Split FashionMNIST. Isso impossibilita afirmar que a adição do cVAE-GAN, por si só, leva a uma melhoria no desempenho de aprendizagem contínua em relação ao treinamento desacoplado com o cVAE original.

5.1.3 Uso de Camadas Convolucionais

Para esse experimento, substituímos todos os módulos da arquitetura por implementações com camadas convolucionais, exceto pelo módulo de classificação, que foi mantido o mesmo da arquitetura original do IRCL. O objetivo é verificar se o uso desse tipo de camada leva a uma melhora no desempenho obtido pela arquitetura. Apesar do desempenho misto apresentado no experimento anterior, optamos por manter a utilização do cVAE-GAN visto que os resultados obtidos com ele ainda foram melhores do que os obtidos pela arquitetura original.

Para isso, utilizamos um módulo específico composto de três camadas convolucionais e ReLU como função de ativação, cada uma seguida por uma operação de *max pooling* e com

uma camada linear de saída. O codificador foi implementado com três camadas convolucionais, aplicando *batch normalization* e a função de ativação ELU, com $\alpha = 1.0$. O decodificador possui uma camada linear de entrada, com ELU como ativação, seguida por três camadas convolucionais com a função de ativação ReLU. Foi aplicado *batch normalization* em todas as suas camadas. O discriminador é composto por quatro camadas convolucionais, usando ReLU como ativação, e uma camada linear de saída, com aplicação de *batch normalization* em todas as camadas escondidas..

A Tabela 5 exibe os hiperparâmetros utilizados durante este experimento para todos os módulos da arquitetura.

Tabela 5 – Hiperparâmetros utilizados

Conjunto de Dados	**Módulo**	**Épocas**	**Taxa de Aprendizado**
Split MNIST	Específico	25	4×10^{-4}
	Classificador		
	Codificador	70	10^{-3}
	Decodificador		
	Discriminador	70	2×10^{-4}
Split FashionMNIST	Específico	15	2×10^{-4}
	Classificador		
	Codificador	70	10^{-3}
	Decodificador		
	Discriminador	70	2×10^{-4}

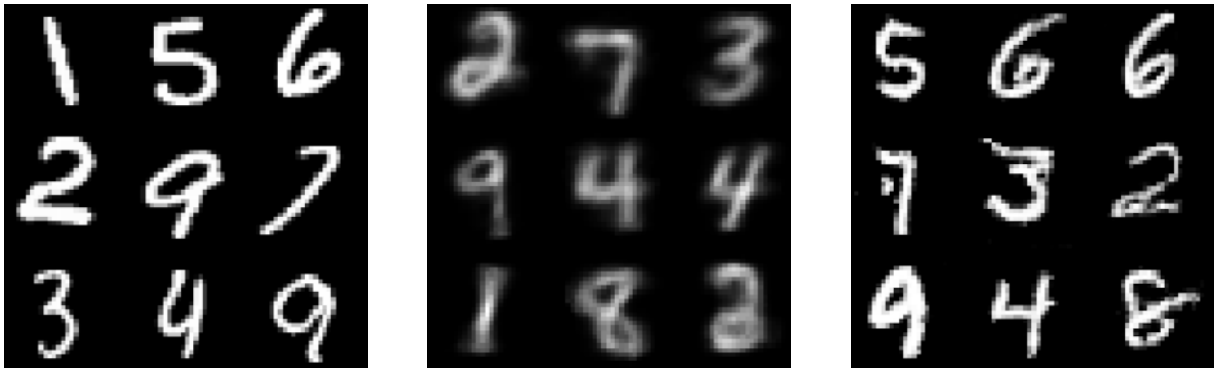
A Tabela 6 apresenta os resultados obtidos, onde é possível verificar que a utilização de camadas convolucionais levou a uma melhora em todas as métricas avaliadas.

Tabela 6 – Comparação das Médias do SSIM, do BWT e da Acurácia

Método de Treino	**Split MNIST**			**Split FashionMNIST**		
	SSIM	BWT (%)	Acurácia (%)	SSIM	BWT (%)	Acurácia (%)
Original	0,599	-11,314	87,143	0,517	-23,494	76,070
Desacoplado	0,643	-11,680	87,227	0,519	-17,887	77,940
Desacoplado cVAE-GAN	0,618	-9,554	88,761	0,547	-21,964	77,160
Desacoplado cVAE-GAN Camadas Convolucionais	**0.785**	**-2.989**	**97.168**	**0.599**	**-15.112**	**81.329**

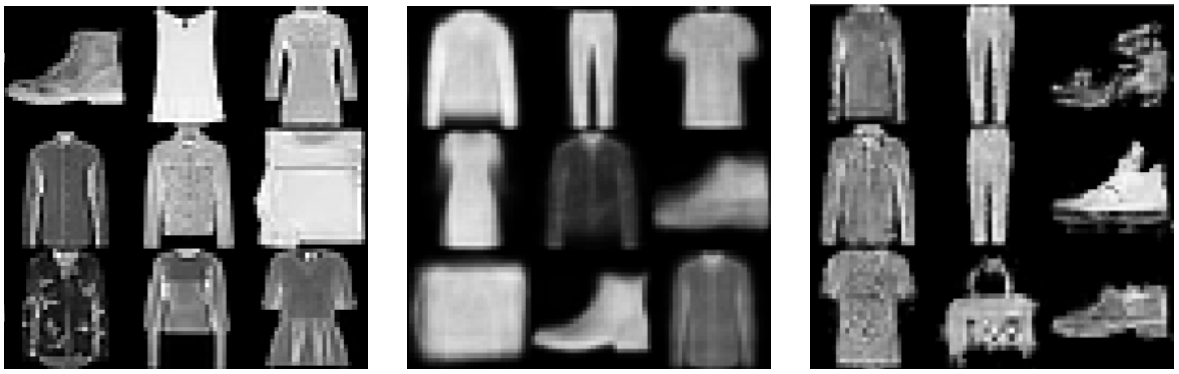
Ao analisar os valores de SSIM, o uso de camadas convolucionais resultou em uma melhora na qualidade das imagens geradas nos dois conjuntos de dados avaliados. As Figuras 20 e 21

apresentam as imagens do MNIST e Fashion-MNIST em conjunto com as imagens geradas utilizando o método IRCL e as produzidas através de nossa abordagem. É possível observar um aumento na nitidez das imagens geradas pela nossa abordagem em comparação as imagens geradas pelo IRCL.



(a) MNIST (b) IRCL (c) Método proposto

Figura 20 – a) Amostras do MNIST em comparação com b) imagens gerada pelo IRCL e c) imagens gerada pelo método proposto



(a) FashionMNIST (b) IRCL (c) Método proposto

Figura 21 – a) Amostras do Fashion-MNIST em comparação com b) imagens gerada pelo IRCL e c) imagens gerada pelo método proposto

Em relação ao BWT, houve uma melhora na taxa de esquecimento, possívelmente uma consequência da maior qualidade das imagens geradas, que são usadas para manter o conhecimento relativo as classes de tarefas anteriores. Finalmente, considerando a acurácia média, foram obtidas as melhores taxas em ambos conjuntos avaliados.

Dessa forma, o experimento demonstrou que a combinação da utilização de camadas convolucionais com o treinamento desacoplado e uso de um cVAE-GAN resulta na diminuição do efeito de esquecimento catastrófico e, consequentemente, na melhora no desempenho de classificação ao final do treinamento de todas as tarefas.

6 CONCLUSÃO

Neste trabalho, propusemos uma arquitetura aprimorada para Aprendizado Contínuo utilizando a abordagem de *pseudo-replay* no cenário de classe incremental. O estudo se baseou no método IRCL (*Invariant Representation for Continual Learning*), considerado o estado da arte até então. A arquitetura proposta integra a utilização de camadas convolucionais com o uso de um cVAE-GAN como modelo gerativo, além de utilizar um treinamento independente para os módulos da arquitetura. Os conjuntos de dados avaliados foram o *Split MNIST* e *Split FashionMNIST*.

Primeiramente, conduzimos experimentos para testar a hipótese de que o desacoplamento do treinamento do modelo generativo dos demais módulos da arquitetura permitiria a utilização de melhores hiperparâmetros. Como resultado, observamos que desacoplar o treinamento leva a uma melhora na qualidade das imagens geradas e aumento no desempenho de classificação das imagens.

Em seguida, avaliamos a hipótese de que o desacoplamento viabilizaria o treinamento de modelos generativos mais robustos, com melhor capacidade de geração de imagens. Para isso, utilizamos um cVAE-GAN no lugar do cVAE original, treinado de forma desacoplada. Os resultados obtidos não foram suficientes para afirmar que o uso do cVAE-GAN leva a uma melhora no desempenho em relação ao uso do cVAE tradicional.

Por fim, realizamos experimentos para testar a hipótese de que o uso de camadas convolucionais levaria a um melhor geral desempenho da arquitetura. Os resultados demonstraram que o uso de camadas convolucionais em conjunto com o treinamento desacoplado e a utilização de um cVAE-GAN leva a uma melhora na qualidade das imagens geradas, um aumento na acurácia da classificação e diminuição do efeito de esquecimento catastrófico.

Dessa forma, a abordagem desenvolvida neste estudo superou o método IRCL usado como referência, sendo possível obter uma melhora de até 10 pontos percentuais na média da Acurácia e de até 8 pontos na média do *Backward Transfer* nos conjuntos avaliados.

REFERÊNCIAS

CIREsAN, D. C.; MEIER, U.; MASCI, J.; GAMBARDELLA, L. M.; SCHMIDHUBER, J. Flexible, high performance convolutional neural networks for image classification. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*. [S.l.]: AAAI Press, 2011. (IJCAI'11), p. 1237–1242. ISBN 9781577355144.

CIREGAN, D.; MEIER, U.; SCHMIDHUBER, J. Multi-column deep neural networks for image classification. In: IEEE. *2012 IEEE conference on computer vision and pattern recognition*. [S.l.], 2012. p. 3642–3649.

COSSU, A.; ZIOSI, M.; LOMONACO, V. Sustainable Artificial Intelligence through Continual Learning. *Proceedings of the 1st International Conference on AI for People: Towards Sustainable AI, CAIP 2021, 20-24 November 2021, Bologna, Italy*, 2021. Disponível em: <https://aiforpeople.org/conference/assets/papers/CAIP21-P11.pdf>.

DAI, Z.; LIU, H.; LE, Q. V.; TAN, M. CoAtNet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, v. 34, p. 3965–3977, 2021.

EBRAHIMI, S.; MEIER, F.; CALANDRA, R.; DARRELL, T.; ROHRBACH, M. Adversarial continual learning. In: VEDALDI, A.; BISCHOF, H.; BROX, T.; FRAHM, J.-M. (Ed.). *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020. p. 386–402. ISBN 978-3-030-58621-8.

FRENCH, R. M. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, Taylor Francis, v. 4, n. 3-4, p. 365–377, 1992. Disponível em: <https://doi.org/10.1080/09540099208946624>.

FRITZKE, B. A growing neural gas network learns topologies. In: *Proceedings of the 7th International Conference on Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1994. (NIPS'94), p. 625–632.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial networks. *Communications of the ACM*, ACM New York, NY, USA, v. 63, n. 11, p. 139–144, 2020.

GOODFELLOW, I. J.; MIRZA, M.; XIAO, D.; COURVILLE, A.; BENGIO, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

HAYKIN, S. *Neural Networks and Learning Machines*. Prentice Hall, 2009. (Neural networks and learning machines, v. 10). ISBN 9780131471399. Disponível em: <https://books.google.com.br/books?id=K7P36IKzI_QC>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2015. p. 1026–1034.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

KIRKPATRICK, J.; PASCANU, R.; RABINOWITZ, N.; VENESS, J.; DESJARDINS, G.; RUSU, A. A.; MILAN, K.; QUAN, J.; RAMALHO, T.; GRABSKA-BARWINSKA, A.; HASSABIS, D.; CLOPATH, C.; KUMARAN, D.; HADSELL, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, v. 114, n. 13, p. 3521–3526, 2017. Disponível em: <https://www.pnas.org/doi/abs/10.1073/pnas.1611835114>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2012.

LANGE, M. D.; ALJUNDI, R.; MASANA, M.; PARISOT, S.; JIA, X.; LEONARDIS, A.; SLABAUGH, G.; TUYTELAARS, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 44, n. 7, p. 3366–3385, 2022.

LARSEN, A. B. L.; SØNDERBY, S. K.; LAROCHELLE, H.; WINTHER, O. Autoencoding beyond pixels using a learned similarity metric. In: PMLR. *International conference on machine learning*. [S.l.], 2016. p. 1558–1566.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, 1989.

LI, C.; ZHUANG, B.; WANG, G.; LIANG, X.; CHANG, X.; YANG, Y. Automated progressive learning for efficient training of vision transformers. In: *CVPR*. [S.l.: s.n.], 2022.

MCCLELLAND, J. L.; MCNAUGHTON, B. L.; O'REILLY, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, American Psychological Association, v. 102, n. 3, p. 419, 1995.

MCCLOSKEY, M.; COHEN, N. J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In: BOWER, G. H. (Ed.). Academic Press, 1989, (Psychology of Learning and Motivation, v. 24). p. 109–165. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0079742108605368>.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MOE-HELGESEN, O.-M.; STRANDEN, H. Catastophic forgetting in neural networks. *Dept. Comput. & Information Sci., Norwegian Univ. Science & Technology (NTNU), Trondheim, Norway, Tech. Rep*, Citeseer, v. 1, p. 22, 2005.

RATCLIFF, R. Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychological Review*, v. 97, n. 2, p. 285–308, 1990. Disponível em: <https://psycnet.apa.org/doi/10.1037/0033-295X.97.2.285>.

REBUFFI, S.-A.; KOLESNIKOV, A.; SPERL, G.; LAMPERT, C. H. icarl: Incremental classifier and representation learning. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 2001–2010.

ROBINS, A. Catastrophic forgetting in neural networks: the role of rehearsal mechanisms. In: *Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*. [S.l.: s.n.], 1993. p. 65–68.

ROBINS, A. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, Taylor Francis, v. 7, n. 2, p. 123–146, 1995. Disponível em: <https://doi.org/10.1080/09540099550039318>.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group UK London, v. 323, n. 6088, p. 533–536, 1986. Disponível em: <https://www.nature.com/articles/323533a0>.

SHIN, H.; LEE, J. K.; KIM, J.; KIM, J. Continual learning with deep generative replay. *Advances in neural information processing systems*, v. 30, 2017.

SOHN, K.; LEE, H.; YAN, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, v. 28, 2015.

SOKAR, G.; MOCANU, D. C.; PECHENIZKIY, M. Learning Invariant Representation for Continual Learning. In: *AAAI-21 International Workshop on Meta-Learning for Computer Vision (MeL4CV), 2021*. [S.l.: s.n.], 2021.

STRUBELL, E.; GANESH, A.; MCCALLUM, A. Energy and Policy Considerations for Modern Deep Learning Research. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 34, n. 09, p. 13693–13696, Apr. 2020. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/7123>.

SYMEONIDIS, G.; NERANTZIS, E.; KAZAKIS, A.; PAPAKOSTAS, G. A. Mlops - definitions, tools and challenges. In: *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.: s.n.], 2022. p. 0453–0460.

TOOSI, A.; BOTTINO, A.; SABOURY, B.; SIEGEL, E. L.; RAHMIM, A. A Brief History of AI: How to Prevent Another Winter (A Critical Review). *CoRR*, abs/2109.01517, 2021. Disponível em: <https://www.pet.theclinics.com/article/S1556-8598(21)00053-5/fulltex>.

VEN, G. M. van de; TOLIAS, A. S. *Three scenarios for continual learning*. 2019.

VERWIMP, E.; LANGE, M. D.; TUYTELAARS, T. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2021. p. 9385–9394.

ZHAI, X.; KOLESNIKOV, A.; HOULSBY, N.; BEYER, L. Scaling vision transformers. *CoRR*, abs/2106.04560, 2021. Disponível em: <https://arxiv.org/abs/2106.04560>.

ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.