Lucas Augusto Mota de Alcantara

Aprendizado contínuo para classificação de imagens do mundo real

Universidade Federal de Pernambuco

Recife
2021

Lucas Augusto Mota de Alcantara

Aprendizado contínuo para classificação de imagens do mundo real

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: *Visão Computacional*
Orientador: *Tsang Ing Ren*

Recife
2021

ABSTRACT

In the context of solving image classification tasks using neural networks, continual learning aims to allow classifiers to be able to learn to classify a sequence of classes, in a way that all the knowledge gathered from previous classes of the sequence is used when learning new ones. The challenge to reach this objective is to avoid the phenomenon known as catastrophic forgetting, which is the tendency of neural networks to lose all the knowledge learned as new data is presented. Current image classification methods based on continuous learning are limited to classifying grayscale images, of low resolution, or from few different classes. This work aims to improve the state-of-the-art classification method based on continuous learning, through the use of more robust models capable of dealing with RGB images, of higher resolution and from a higher number of classes. Our results shows that our approach wasn't able to improve the classification performance on the evaluated datasets, although it was able to improve the forgetting metric. This shows that the changes we propose can have a positive impact on the performance of the architecture and that there is room to explore more deeply each one of them.

Keywords: continual learning, image classification, catastrophic forgetting

RESUMO

No contexto de classificação de imagens utilizando modelos de redes neurais, aprendizado contínuo tem como objetivo permitir que os classificadores sejam capazes de aprender a classificar uma sequência de classes, de forma que o conhecimento obtido no passado seja aproveitado no aprendizado de classes futuras. O desafio para alcançar este objetivo é evitar o fenômeno conhecido como esquecimento catastrófico, que é a tendência que as redes neurais têm de ”esquecer” informações aprendidas quando novos dados são apresentados. Métodos atuais de classificação de imagens baseados em aprendizado contínuo se limitam a classificar imagens em escala de cinza, com baixa resolução e com pouca variedade de classes. Este trabalho tem o objetivo de aprimorar um dos métodos existentes para classificação baseado em aprendizado contínuo, por meio da utilização de arquiteturas mais robustas e capazes de lidar com imagens coloridas, com maior resolução e de uma maior quantidade de classes. Os resultados obtidos mostraram que a as alterações propostas não foram capazes de melhorar o desempenho da classificação dos *datasets* avaliados. Apesar disso, a arquitetura proposta foi capaz de obter resultados melhores em uma das métricas utilizadas para avaliação, o que indica que as alterações propostas podem sim ter impacto positivo no desempenho e que existe espaço para investigar mais a fundo cada uma delas.

Palavras-chave: aprendizado continuo, classificação de imagens, esquecimento catastrófico

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Métodos tradicionais de aprendizagem de máquina supervisionada, especialmente métodos de classificação baseados em redes neurais, geralmente assumem que os dados são independentes e identicamente distribuídos (i.i.d). Porém, em aplicações reais essas condições nem sempre são satisfeitas. Em alguns casos, os dados podem ser acessados apenas de maneira sequencial, vindos de um ambiente dinâmico e em constante alteração [1]. Nesses cenários, métodos tradicionais não são capazes de reter o conhecimento obtido com elementos anteriores da sequência, devido ao fenômeno conhecido na literatura [2] como esquecimento catastrófico (*catastrophic forgetting*) ou interferência catastrófica (*catastrophic interference*). Esse fenômeno é causado devido ao fato de que, ao receber novas amostras de dados, os pesos dos modelos são atualizados para que se adaptem às novas amostras, sem levar em consideração dados antigos.

Com isso, a tendência é que o conhecimento adquirido previamente seja perdido. Ou seja, após ter aprendido um determinado conjunto de padrões, se novos padrões forem apresentados, a rede irá esquecer o conhecimento acumulado até o momento e o substituirá pelo conhecimento obtido a partir dos novos dados. Dessa forma, para que a rede seja capaz de identificar padrões novos e antigos, é necessário que o treinamento seja refeito do zero, dessa vez apresentando à rede os dados de todos os padrões conhecidos até o momento. No contexto de classificação de imagens, os padrões representam as classes de imagens.

O aprendizado contínuo [3] é a subárea de aprendizagem de máquina que estuda técnicas, algoritmos e arquiteturas com o objetivo de evitar ou diminuir o fenômeno de esquecimento catastrófico e, com isso, permitir que o conhecimento adquirido seja acumulado e até mesmo utilizado como base para o aprendizado de novas informações. Isso permite a construção de sistemas dinâmicos, capazes de se adaptar a uma maior variedade de ambientes, aprendendo a resolver novos desafios conforme novos dados são apresentados e sem esquecer como resolver os desafios antigos.

No contexto de classificação de imagens, vários métodos baseados em aprendizado contínuo foram propostos nos últimos anos [4] [5] [6] [7]. Apesar disso, esses métodos não apresentam bons resultados quando aplicados na resolução de problemas reais, que utilizam

imagens coloridas, com maior resolução e com uma quantidade maior de classes.

1.2 Objetivos

O objetivo deste trabalho é aprimorar uma arquitetura proposta na literatura, para melhorar os resultados obtidos em conjuntos de dados que possuam imagens de maior resolução ou com maior número de classes, de forma que seja possível a sua utilização na solução de uma categoria mais abrangente de problemas.

1.3 Estrutura

Este trabalho é dividido da seguinte forma: o Capítulo 2 apresenta alguns conceitos fundamentais da área de aprendizado contínuo, necessários para entender os demais capítulos. O Capítulo 3 apresenta os trabalhos relacionados, incluindo uma descrição do método no qual este trabalho se baseia. No Capítulo 4 é descrito o método proposto. O Capítulo 5 descreve os experimentos realizados para a avaliação do método proposto e apresenta os resultados obtidos. Por fim, o Capítulo 6 apresenta as conclusões.

2

CONCEITOS BÁSICOS

Nesta seção serão apresentados conceitos básicos sobre aprendizado contínuo que serão necessários para o entendimento dos demais capítulos deste trabalho. O foco será nos conceitos relacionados à utilização de aprendizado contínuo em problemas de classificação de imagens, conforme o escopo deste trabalho.

2.1 Definição

Dada uma sequência de conjuntos de dados $s = \{t_0, t_1, ..., t_{N-1}, t_N\}$, onde N é a quantidade de conjuntos, um algoritmo baseado em aprendizado contínuo deve ser capaz de aprender os padrões contidos nessa sequência tendo acesso a apenas um de seus elementos por vez. Ou seja, ao acessar o subconjunto t_i, o método precisa ter aprendido os padrões presentes nos subconjuntos $t_{<i}$. Cada subconjunto t_i da sequência s é chamado de tarefa (do inglês *task*). No caso de problemas de classificação, cada tarefa possui dados pertencentes a um determinado número de classes.

2.2 Cenários

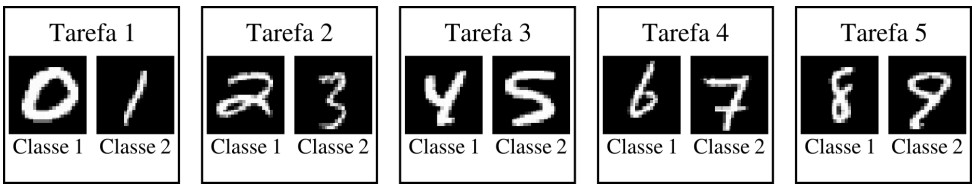


Figura 1: *Dataset* Split-MNIST

Os problemas de aprendizado contínuo podem ser divididos em três cenários [8] de acordo com a disponibilidade do identificador da tarefa para fazer a classificação ou de acordo com a necessidade de inferir ou não o identificador, quando ele não é conhecido. O identificador da tarefa informa a qual subconjunto t_i pertencem os dados a serem classificados. Em todos os cenários, o modelo tem acesso as tarefas de maneira sequencial durante o treinamento. A Figura

1 representa o *dataset* Split-MNIST, uma variação do MNIST tradicional utilizada na avaliação de arquiteturas de aprendizado contínuo, onde ele é dividido em 5 tarefas de 2 classes cada. Ele será utilizado para ilustrar cada um dos cenários.

O primeiro cenário é o de Tarefa Incremental, onde o identificador da tarefa é conhecido. Ter essa informação, faz com que esse seja o cenário mais simples de ser resolvido, pois as opções de classes são reduzidas a quantidade presente em uma única tarefa. É possível, por exemplo, definir modelos diferentes e independentes para cada uma das tarefas e usar o identificador fornecido para selecionar o classificador a ser utilizado. Nesse caso, considerando a Figura 1, como o modelo já sabe a qual tarefa pertence a imagem, basta classificar entre Classe 1 ou Classe 2, ou entre os dígitos 2 e 3, por exemplo, no caso da tarefa 2.

No segundo cenário, chamado de Domínio Incremental, o identificador da tarefa não é fornecido para fazer a classificação. Apesar disso, nesse caso o classificador precisa inferir apenas o identificador da classe, sem indicar a qual tarefa ela pertence. Por exemplo, no Split-MNIST o classificador indicaria se a imagem pertence a alguma das Classes 1 (0, 2, 4, 6, 8) ou a alguma das Classes 2 (1, 3, 5, 7, 9).

O terceiro cenário é o de Classe Incremental e é o mais desafiador, pois nele o identificador da tarefa não é fornecido e deve ser inferido pelo classificador. Ou seja, o modelo precisa indicar exatamente a qual classe de toda a sequência de tarefas pertence a imagem. No caso do Split-MNIST, o modelo retornaria o dígito exato representado na imagem de entrada.

2.3 Abordagens

Os métodos atuais para se conseguir realizar aprendizado contínuo podem ser divididos em três categorias principais [3] [9], sendo elas os métodos baseados em regularização, baseados em arquitetura dinâmica e os métodos baseados em repetição. As diferenças entre cada um deles está na forma como as informações de cada tarefa são acumuladas durante o aprendizado.

A regularização é a adição de um componente na função de perda, que tem o objetivo de evitar que o modelo apresente *overfitting* para o conjunto de dados de treinamento. Os métodos baseados em regularização adicionam um componente na função de perda para que, durante o aprendizado de novas tarefas, alterações realizadas em pesos da rede que são importantes para o desempenho das tarefas já aprendidas sejam penalizadas. Ou seja, a regularização evita que ocorra o *overfitting* para os dados das novas tarefas. Ela diminui a modificação do conhecimento adquirido anteriormente quando novas tarefas precisam ser aprendidas.

Métodos baseados em arquitetura dinâmica modificam a arquitetura do modelo para evitar que o aprendizado de uma nova tarefa interfira no conhecimento aprendido com tarefas anteriores. Isso pode ser feito instanciando uma rede para cada tarefa, dessa forma o modelo treinado com os dados de uma tarefa não é modificado com a chegada de outras. Essa abordagem geralmente necessita que o identificador da tarefa seja fornecido durante a inferência como no cenário de Tarefa Incremental, pois dessa forma é possível definir a rede a ser utilizada.

Os métodos baseados em repetição armazenam uma parte das imagens de tarefas anteriores para utilizá-los durante o treinamento de novas tarefas. Dessa forma, o modelo sempre é treinado com imagens de todas as tarefas, mesmo que em proporções diferentes, o que diminui o esquecimento das tarefas mais antigas.

Uma subcategoria dos métodos baseados em repetição são os métodos baseados em pseudo-repetição. Nesse caso, ao invés de armazenar as imagens, são utilizados modelos generativos para produzir novas imagens que representem imagens de tarefas anteriores. Por conta disso, a qualidade das imagens produzidas pelos modelos generativos tem grande importância na manutenção do conhecimento acumulado até o momento [7]. É em um método dessa subcategoria que este trabalho se baseia.

2.4 Modelos Generativos

Modelos Generativos são redes capazes de produzir amostras de dados artificiais, no nosso caso, modelos capazes de gerar imagens. Nos últimos anos, esse tipo de rede vem recebendo cada vez mais atenção devido a sua capacidade de gerar imagens com um alto grau de realismo. Existem vários tipos de modelos generativos [10], nesta seção iremos focar em dois deles, os Autoencoders Variacionais [11] e as GANs [12].

2.4.1 Autoencoders Variacionais

(a) LBAE (b) NVAE (c) VQ-VAE2

Figura 2: Imagens geradas por *Autoencoders* Variacionais

Os *Autoencoders* Variacionais são um tipo de *autoencoder* onde a distribuição da sua representação intermediária é regularizada durante o treinamento, de forma a garantir que ela tenha um formato conhecido. Com isso, é possível gerar novas representações, seguindo a distribuição definida, e fornecê-las ao *decoder* que, por sua vez, irá produzir as imagens correspondentes. A Figura 2 mostra imagens artificialmente criadas por Autoencoders Variacionais, sendo eles o LBAE [13] na Figura 2a, NVAE [14] na Figura 2b e o VQ-VAE2 [15] na Figura 2c.

2.4.2 GANs

(a) BigGAN (b) StyleGAN

Figura 3: Imagens geradas por Redes Generativas Adversárias

As GANs, ou Redes Generativas Adversárias, são um tipo de arquitetura que possui, pelo menos, duas redes que competem entre si. A primeira delas é chamada de geradora e é responsável por construir imagens. A segunda rede é chamada de discriminadora e é responsável por classificar as imagens produzidas pela rede geradora entre imagens reais e artificiais. As duas redes são treinadas em conjunto, de maneira que a rede geradora está sempre tentando gerar imagens mais verossímeis a fim de enganar a rede discriminadora, enquanto esta está sempre tentando melhorar a sua capacidade de discriminação. Dessa forma, as duas redes competem entre si e evoluem nesse processo, tendo como resultado a geração de imagens cada vez mais reais. É por conta disso que elas recebem o nome de redes ¨Adversárias¨. A Figura 3 mostra exemplos de imagens produzidas por duas GANs, sendo elas a BigGAN [16] na Figura 3a e a StyleGAN [17] na Figura 3b.

3

TRABALHOS RELACIONADOS

Nos últimos anos, o aprendizado contínuo vem cada vez mais ganhando interesse pois, com o aumento da quantidade de dados disponíveis e o crescimento dos modelos de redes neurais, o custo para armazenar esses dados e treinar essas arquiteturas é cada vez maior. Por conta disso, treinar um modelo do zero e com todos os dados sempre que se deseja adicionar uma nova classe a ele não é viável. Nesta seção serão apresentados os trabalhos relacionados baseados nas três abordagens citadas no capítulo anterior.

3.1 Métodos baseados em Regularização

Learning without Forgetting (LwF) [18] propõe que os valores de saída fornecidos pela rede ao receber dados de novas tarefas sejam armazenados antes que seja iniciado o processo de treinamento. Como componente de regularização, é adicionado à função de perda a função *Knowledge Distillation* [19]. Essa função tem o objetivo de induzir que as saídas de uma rede se aproximem das saídas de uma outra. Nesse caso, ela é utilizada para que, durante o treinamento de novas tarefas, os valores de saída apresentados pelos neurônios correspondentes às classes de tarefas anteriores se mantenham próximos aos valores que foram armazenados antes do treinamento.

Em *Elastic Weight Consolidation* (EWC) [20] a alteração dos pesos da rede durante o treinamento de uma nova tarefa é inversamente proporcional à importância deles para o desempenho da rede em tarefas anteriores. A importância de cada um dos parâmetros é obtida a partir da matriz de informação de Fisher, calculada a partir de uma aproximação da função de densidade dos parâmetros à uma distribuição normal, feita utilizando o método de Laplace.

3.2 Métodos baseados em Arquitetura Dinâmica

Progressive Neural Networks (PNN) [21] cria uma nova rede para cada nova tarefa. Para que o conhecimento obtido com tarefas anteriores seja aproveitado, são criadas conexões entre as camadas das redes, de forma que o treinamento de uma rede não afete a outra. PackNet [22] atribui um subconjunto de pesos a cada tarefa a ser treinada. Para isso, o treinamento de uma

tarefa é dividido em duas partes, na primeira delas a rede é treinada sem alterar os subconjuntos de parâmetros das tarefas anteriores. Após isso, os parâmetros com maior magnitude são selecionados para formar o subconjunto da tarefa atual, enquanto os demais parâmetros serão utilizados para o treinamento das próximas tarefas. Na segunda etapa do treinamento, o subconjunto de parâmetros selecionado é treinado novamente com os dados da tarefa atual. Dessa forma, o desempenho em cada tarefa é preservado.

3.3 Métodos baseados em Repetição

O método proposto em *Incremental Classifier and Representation Learning* (iCaRL) [23] armazena um subconjunto das imagens de classes anteriores e o reutiliza durante o treinamento das novas tarefas. As imagens selecionadas para o subconjunto são aquelas cujos vetores de representação mais se aproximam do vetor médio de cada classe, isso é feito utilizando o método *nearest centroid classifier*. Em *Continual Learning with Experience And Replay* (CLEAR) [24] o subconjunto de imagens é criado utilizando o método de *reservoir sampling*, de forma que o subconjunto tenha um tamanho limitado. *Continual Prototype Evolution* (CoPE) [25] utiliza uma abordagem que combina os métodos *nearest centroid classifier* e *reservoir sampling*.

Em *Deep Generative Replay* (DGR) [7] são utilizados dois modelos, onde o primeiro deles é uma GAN, responsável por gerar imagens de classes anteriores, e o outro é o modelo responsável por fazer a classificação. A cada nova tarefa são criados novos pares de gerador e classificador, de forma que o gerador da tarefa anterior fornece imagens para o modelo da tarefa atual. Essa abordagem foi denominada de ¨modelo escolar¨ (*scholar model*), pois o modelo da tarefa anterior representa um professor, enquanto o modelo da tarefa atual representa o estudante que aprende com o professor.

Pseudo-Rehearsal Embedding Regularization (PRER) [5] utiliza um modelo generativo para criar novos vetores de representação e, a partir disso, gerar as imagens correspondentes. Além disso, é utilizada uma regularização para induzir que os vetores de representação se aproximem dos vetores das imagens reais das classes anteriores.

Learning Invariant Representation for Continual Learning (IRCL) [4], método no qual este trabalho se baseia, propõe a arquitetura ilustrada na Figura 4, composta de três componentes: um *autoencoder* variacional condicional (CVAE), um módulo específico e um classificador. Durante o treinamento das tarefas, o módulo especifico recebe a imagem de entrada e produz uma representação características discriminativas em relação as classes, enquanto o *encoder* produz uma representação invariante, ou seja, sem características exclusivas das classes. As duas representações são combinadas e utilizadas como entrada do classificador.

A divisão da representação em uma parte discriminativa e outra invariante foi proposta em *Adversarial Continual Learning* [26], onde os autores mostraram que a parte invariante sofre menos com o efeito de esquecimento catastrófico e que a sua utilização tem efeito positivo no aprendizado contínuo.

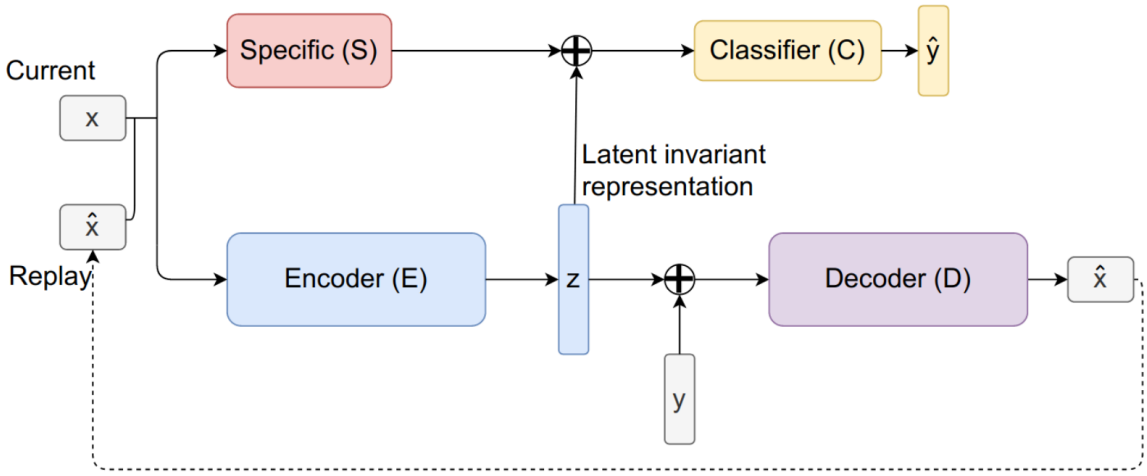


Figura 4: Arquitetura proposta em *Learning Invariant Representation for Continual Learning*

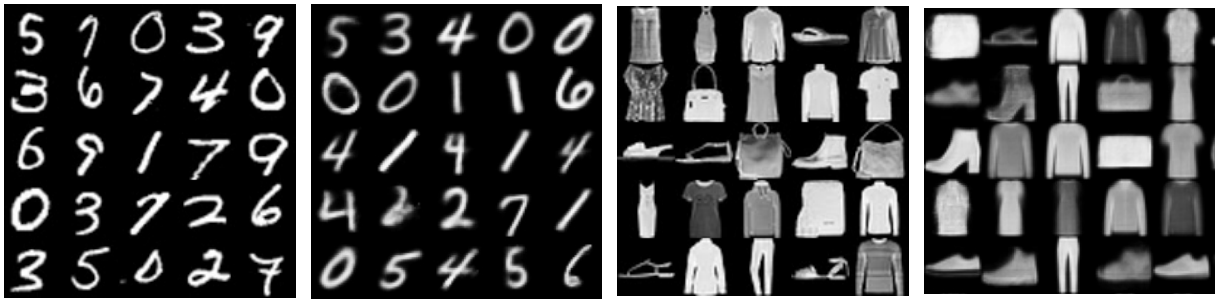
Para induzir o *encoder* a gerar uma representação que não seja discriminativa, é adicionada à sua saída a *label* y que identifica a classe da imagem de entrada x. Como mostrado em *A Cyclically-Trained Adversarial Network for Invariant Representation Learning* [27] e *Adversarial Autoencoders* [28], adicionar essa informação na representação produzida pelo *encoder* reduz a presença de características específicas enquanto mantém informações independentes da classe. Além disso, adicionar a *label* na representação faz com que a geração de imagens seja condicional, ou seja, permite escolher exatamente a classe da imagem a ser gerada. A identificação da classe, y, é codificada utilizando one-hot encoding e concatenada ao vetor de representação z.

Por ser invariante, a representação intermediária do *autoencoder* não pode ser utilizada sozinha para fazer a classificação. Por conta disso, é adicionado o módulo específico, responsável por obter características discriminativas. Assim, as duas representações são combinadas para formar a entrada do classificador.

A geração de imagens é feita pelo *autoencoder*, que possui um componente de regularização na sua função de perda para induzir o *encoder* a gerar representações intermediárias cujo espaço siga uma distribuição normal, através do cálculo da Divergência de *Kullback-Leibler* [11]. As imagens geradas são utilizadas durante o treinamento de novas tarefas.

A Equação 3.1 representa a função de perda utilizada pelo CVAE. O primeiro termo representa o erro de reconstrução das imagens, calculado através do Erro Quadrático Médio (MSE), enquanto o segundo termo é a Divergência de *Kullback-Leibler*. A imagem de entrada é representada por x, enquanto $\hat{x}$ representa a imagem reconstruída. A distribuição aprendida pelo *encoder* é dada por $q(z|x)$, $p(z)$ é a distribuição a ser aproximada pelo *encoder*, nesse caso uma normal.

$$\mathcal{L}_{CVAE} = \|x - \hat{x}\|^2 - KL(q(z|x)\|p(z)) \tag{3.1}$$



(a) Imagens originais MNIST (b) Imagens geradas MNIST (c) Imagens originais *Fashion*-MNIST (d) Imagens geradas *Fashion*-MNIST

Figura 5: Exemplos de imagens originais e geradas a partir dos *datasets* MNIST e *Fashion*-MNIST

O método proposto foi capaz de superar o estado da arte de métodos baseados em pseudo-repetição nos *datasets* MNIST e *Fashion*-MNIST, obtendo 86% e 76% de acurácia, respectivamente. Os *datasets* foram divididos em 5 tarefas, cada uma delas com 2 classes. A Figura 5 contém exemplos de imagens dos *datasets* utilizados, assim como exemplos de imagens geradas pelo autoencoder durante o treinamento.

4

MÉTODO PROPOSTO

Como base para este trabalho, escolhemos usar a abordagem apresentada em IRCL pois, até o momento, esse é o método que apresenta os melhores resultados em aprendizado contínuo baseado em repetição. Neste Capítulo serão apresentadas as alterações, e suas motivações, propostas para a implementação original do IRCL.

4.1 *Invariant Representation for Continual Learning*

A arquitetura do método IRLC possui uma implementação oficial disponível[1] na qual as redes do modelo foram construídas utilizando camadas *fully-connected*, um tipo de camada que não é ideal para ser utilizada em redes para processamento de imagens [29]. Ao observar a Figura 5 no Capítulo 3, é possível perceber que, para o MNIST e *Fashion*-MNIST, o *autoencoder* variacional utilizado foi capaz de gerar imagens com formas bem definidas, mas que não possuem a mesma nitidez das imagens originais. Isso fica mais claro ao analisar a Figura 6, onde o modelo foi utilizado para reconstruir e gerar imagens do CIFAR-10, um *dataset* com imagens mais detalhadas e com maior resolução (32x32) do que o MNIST ou *Fashion*-MNIST (28x28).

(a) Imagens originais (b) Imagens Reconstruídas (c) Imagens geradas

Figura 6: Exemplos de imagens originais e produzidas pelo autoencoder a partir do CIFAR-10

[1]Learning Invariant Representation for Continual Learning `https://github.com/GhadaSokar/Invariant-Representation-for-Continual-Learning`

Ao utilizar o *dataset* EMNIST [30], uma extensão do MNIST que adiciona imagens de letras, pela Figura 7 é possível verificar que o autoencoder utilizado também não consegue manter a qualidade das imagens quando o número de classes aumenta. Nesse teste foram utilizadas 26 letras, em maiúsculo, divididas em 13 tarefas com 2 classes cada. As imagens da Figura 7 foram geradas ao final do treinamento de todas as tarefas, de forma que as imagens de entrada contém apenas as letras Y e Z, que são as letras presentes na última tarefa. A Figura 7a contém exemplos das imagens de entrada da tarefa 13, a Figura 7b mostra as imagens reconstruídas pelo *autoencoder* a partir das imagens da Figura 7a, enquanto a Figura 7c possui imagens geradas pelo modelo a partir de todas as classes aprendidas.

(a) Imagens originais (b) Imagens reconstruídas (c) Imagens geradas

Figura 7: Exemplos de imagens originais e geradas a partir do *dataset* EMNIST

Baseado nessas observações, nós propomos a utilização de um modelo generativo mais robusto, que seja capaz de gerar imagens com maior nível de detalhes, além de utilizar camadas convolucionais tanto no modelo generativo quanto no módulo específico.

4.2 Modelo Generativo

O fato de o espaço das representações intermediárias de *autoencoders* variacionais tradicionais (VAEs) ser induzido a seguir uma distribuição normal é um dos fatores que faz com que as imagens geradas tenham pouca definição [31] [32]. Por conta disso, escolhemos como autoencoder o *Latent Bernoulli Autoencoder* (LBAE) [13], cuja representação intermediária está no espaço de Bernoulli, ou seja, o vetor de representação intermediária possui apenas valores binários. Além disso, o *encoder* não é induzido a produzir representações seguindo nenhuma distribuição específica. Essas características fazem com que seja possível reconstruir e gerar imagens com maior qualidade, como mostram as Figuras 8 e 9.

Como mostrado na Figura 2 no Capítulo 2, os *autoencoders* NVAE e VQ-VAE2 conseguem gerar imagens visualmente melhores do que aquelas produzidas pelo LBAE. Porém, esses modelos têm um custo computacional muito maior, de forma que a sua utilização para esse projeto se torna inviável dado o *hardware* e o tempo disponíveis. O NVAE, por exemplo,

(a) MNIST (b) CIFAR-10 (c) CelebA

Figura 8: Exemplos de imagens originais e reconstruídas pelo LBAE

(a) MNIST (b) CIFAR-10 (c) CelebA

Figura 9: Exemplos de imagens geradas pelo LBAE

precisa de 21 horas para ser treinado no MNIST, utilizando duas *GPUs* V100 com 16GB de *VRAM* cada, de acordo com o seu repositório oficial[2]. Para o CIFAR-10, o tempo é de 55 horas utilizando oito V100. O VQ-VAE2, por sua vez, não possui uma implementação oficial disponível e as implementações não oficiais não possuem a parte de geração condicional de imagens implementada[3].

A Figura 10 ilustra a arquitetura do LBAE, incluindo o processo de binarização da representação produzida pelo *encoder*. Para binarizar uma representação intermediária com N dimensões $z \in \mathbb{R}^N$, os autores utilizaram a Função 4.1 para aplicar um limiar em cada posição do vetor de representação.

$$f_b(z_i) = \begin{cases} 1, \text{ se } z_i \geq 0 \\ \text{-1, caso contrário.} \end{cases} \tag{4.1}$$

[2]Running the main NVAE training and evaluation scripts `https://github.com/NVlabs/NVAE#running-the-main-nvae-training-and-evaluation-scripts`

[3]Conditioned Sample #9 `https://github.com/rosinality/vq-vae-2-pytorch/issues/9#issuecomment-511283100`

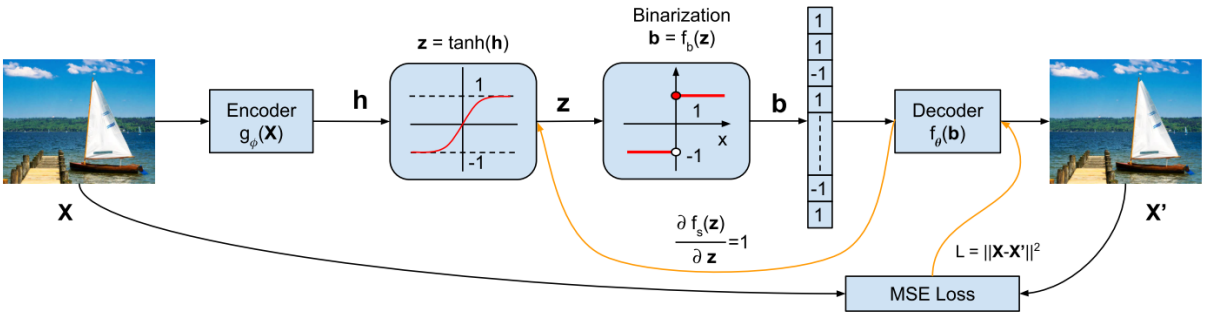


Figura 10: Arquitetura do *Latent Bernouli Autoencoder*

Como essa função possui uma descontinuidade e, portanto, não é diferenciável, os autores adicionaram uma função $f_s(z) = z$, com gradiente $\nabla_z f_s = 1$, para permitir o *backpropagation*, de forma que o gradiente passe do *decoder* diretamente para o *encoder* sem considerar a binarização.

Com a utilização do LBAE, nós implementamos o módulo específico de forma que ele gere uma representação no mesmo espaço de Bernoulli usado pelo LBAE, para que as representações específica e invariante estejam no mesmo espaço. Para isso, assim como é feito no *encoder* do LBAE, nós utilizamos a função de ativação *Tanh* na ultima camada do módulo específico e aplicamos a função f_b para binarizar a representação.

Originalmente, o LBAE não é capaz de gerar imagens de forma condicional, ou seja, não é possível definir a classe das imagens a serem geradas. Porém, o método proposto em IRCL se baseia em um modelo generativo condicional, capaz de gerar representações intermediárias invariantes. Para fazer o LBAE produzir o mesmo tipo de representação, nós concatenamos à representação gerada pelo *encoder* um vetor que contém a informação da classe da imagem a ser criada. Essa informação foi codificada usando *one-hot encoding*, assim como é feito na implementação do IRCL. No entanto, a codificação também foi feita no espaço de Bernoulli usado pelo LBAE, ou seja, as posições que normalmente seriam preenchidas com 0 agora são preenchidas com -1, enquanto a posição que teria o valor 1 continua recebendo esse valor. Com a adição da classe da imagem na representação interediária, o LBAE passa a ser um modelo condicional, ao qual iremos nos referir a partir daqui como CLBAE.

Como o CLBAE não utiliza nenhum componente de regularização na sua função de perda, ela é definida apenas pelo erro da reconstrução da imagem, MSE, como mostra a Equação 4.2.

$$\mathcal{L}_{CLBAE} = \|x - \hat{x}\|^2 \tag{4.2}$$

Por último, baseado na forma como as GANs são treinadas, nós propomos a adição de um discriminador para avaliar as imagens que são produzidas pelo *decoder*, formando um VAE/GAN, ou CLBAE/GAN, como foi originalmente proposto em *Autoencoding beyond pixels using a learned similarity metric* [33]. O discriminador deve competir com o autoencoder e fazer

com que ele produza imagens mais próximas das reais. A arquitetura proposta é ilustrada na Figura 11.

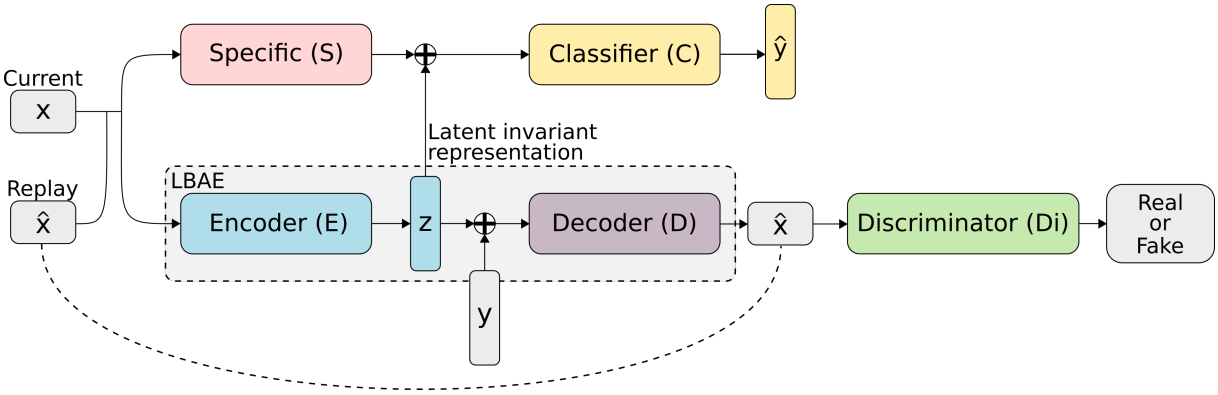


Figura 11: Arquitetura proposta

5

EXPERIMENTOS E RESULTADOS

Para avaliar o impacto das alterações propostas, realizamos experimentos utilizando os *datasets* MNIST e *Fashion*-MNIST, assim como o artigo original do IRCL, e também o EMNIST e CIFAR-10 para a verificar o desempenho quando o conjunto de dados possui mais classes ou imagens com mais detalhes. Os datasets foram organizados em tarefas com 2 classes cada, assim como no artigo do IRCL. Dessa forma, o MNIST, *Fashion-MNIST* e CIFAR-10 foram divididos em 5 tarefas, enquanto o EMNIST foi dividido em 13. Os resultados obtidos foram comparados com aqueles apresentados pelo método original.

Utilizamos para avaliação as duas métricas usadas no artigo do IRCL, a primeira delas é a média das acurácias em cada tarefa obtida ao final do treinamento de todas elas, definida na Equação 5.1. A segunda foi o *Backward Transfer* (BWT), definida na Equação 5.2, que foi proposta em *Gradient Episodic Memory for Continual Learning* [34], e avalia o quanto o aprendizado de uma nova tarefa t afeta o desempenho do modelo em tarefas anteriores $k < t$.

$$Acc_{avg} = \frac{1}{T}\sum_{n=1}^{T} Acc_{T,i} \tag{5.1}$$

$$BWT = \frac{1}{T-1}\sum_{n=1}^{T-1} (Acc_{T,i} - Acc_{i,i}) \tag{5.2}$$

Um valor positivo de BWT indica que o aprendizado de novas tarefas melhoram o desempenho obtido em tarefas anteriores. Da mesma forma, valores negativos de BWT indicam que o desempenho apresentado pelo modelo em tarefas anteriores foi prejudicado ao aprender novas tarefas. Ou seja, valores muito negativos de BWT são um indicativo de que está acontecendo o esquecimento catastrófico durante o treinamento. O termo $Acc_{j,i}$ nas Equações 5.1 e 5.2 representa a acurácia na tarefa i após o aprendizado da tarefa j, sendo $j > i$, e T representa o número total de tarefas.

5.1 Experimentos

Para estabelecer um *baseline* para comparação, treinamos a arquitetura original do IRCL 5 vezes em cada *dataset*, por 5, 10 e 20 épocas para cada tarefa. Após isso, foi feita a média dos experimentos para cada quantidade de épocas. Os hiperparâmetros utilizados foram os mesmos da implementação original, onde foi utilizado o Adam como otimizador, 0.01 como taxa de aprendizagem do autoencoder, enquanto a do módulo específico e do classificador foram de 0.0002 e o tamanho de batch foi 128. Após o treinamento de cada tarefa, são geradas 5000 imagens das classes de tarefas anteriores para serem utilizadas durante o treinamento da próxima tarefa.

O tamanho dos vetores de representação específica e invariante para o MNIST e *Fashion-MNIST* foram os mesmos usados na implementação original do IRCL, ou seja, 20 para a específica e 32 para a invariante, para os dois *datasets*. Para o EMNIST e CIFAR-10, definimos o vetor invariante com tamanho 48 e 64, respectivamente. Para o vetor de representação específica os tamanhos foram de 30 para o EMNIST e 40 para o CIFAR-10.

Para avaliar a nossa arquitetura, para cada *dataset* nós também treinamos o modelo 5 vezes por 5, 10 e 20 épocas em cada tarefa e calculamos a média. Esses experimentos foram realizados com e sem o discriminador para avaliar as imagens geradas, para que seja verificada a influência desse componente no desempenho. O otimizador utilizado foi o Adam, com uma taxa de aprendizado de 0.0001 para todos os componentes da arquitetura, e 128 como tamanho do *batch*. Ao final do treinamento de uma tarefa, 5000 imagens das classes de tarefas anteriores são geradas para o treinamento da próxima.

O tamanho definido para as representações produzidas pelo *encoder* e pelo módulo especifico foi de 200 para o MNIST, 300 para o EMNIST e *Fashion*-MNIST e 600 para o CIFAR-10. Para o MNIST e CIFAR-10 os tamanhos definidos foram os mesmos usados no artigo do LBAE.

5.2 Resultados

As tabelas apresentadas nessa seção contém as médias das acurácias e BWTs obtidas em cada conjunto de 5 execuções de 5, 10 e 20 épocas. As Equações 5.3 e 5.4 representam as métricas que são apresentadas nas tabelas, onde Acc_{avgi} e BWT_i representam, respectivamente, a acurácia e o BWT calculados na i-ésima execução a partir das Equações 5.1 e 5.2.

$$Acc_{table} = \frac{1}{5}\sum_{n=1}^{5} Acc_{avgi} \tag{5.3}$$

$$BWT_{table} = \frac{1}{5}\sum_{n=1}^{5} BWT_i \tag{5.4}$$

Foram incluidos os resultados conseguidos pela arquitetura original proposta no artigo do IRCL e também os resultados obtidos a partir das alterações propostas nesse trabalho, incluindo resultados conseguidos com e sem a presença do discriminador na nossa arquitetura.

5.2.1 MNIST

A Tabela 1 apresenta os resultados obtidos para o *dataset* MNIST. A partir dela, é possível verificar que o nosso modelo não foi capaz de melhorar os resultados da arquitetura original do IRCL. No entanto, pode ser observado na Tabela 1a que a adição do componente discriminador melhorou a acurácia quando do modelo quando ele foi treinado por 5 e 10 épocas, em relação aos resultados da arquitetura sem esse componente. Além disso, na Tabela 1b é possível ver que adicionar o discriminador melhorou o BWT obtido quando o modelo foi treinado por 10 e 20 épocas, quando comparado à arquitetura sem discriminador.

	5 épocas	10 épocas	20 épocas
IRCL Original	84,882%	**85,374%**	84,766%
Nosso (sem Disc.)	74,271%	75,624%	72,897%
Nosso (com Disc.)	75,144%	80,429%	70,802%

(a) Acurácia média

	5 épocas	10 épocas	20 épocas
IRCL Original	**-13,172%**	-13,820%	-14,876%
Nosso (sem Disc.)	-17,314%	-18,580%	-25,534%
Nosso (com Disc.)	-18,291%	-17,714%	-19,566%

(b) BWT médio

Tabela 1: Resultados obtidos no MNIST

Na Figura 12 são apresentadas as imagens geradas pelos modelos após o treinamento da última tarefa do MNIST por 20 épocas. É perceptível que os nossos modelos foram capazes de gerar imagens com maior definição do que aquelas produzidas pela implementação original do IRCL. Apesar disso, todos os dígitos gerados pela arquitetura original, presentes na Figura 12a, são facilmente identificáveis, enquanto os nossos modelos geram algumas imagens que são mais difíceis de reconhecer, como visto nas Figuras 12b e 12c.

(a) IRCL Original

(b) Nosso (sem Disc.)

(c) Nosso (com Disc.)

Figura 12: Imagens do MNIST geradas pelos modelos treinados por 20 épocas

5.2.2 *Fashion*-MNIST

Na Tabela 2 estão os resultados para o *dataset Fashion*-MNIST. Aqui, a arquitetura original do IRCL também obteve os melhores resultados de acurácia mas, ao contrário do *dataset* anterior, a arquitetura com discriminador foi a que apresentou o melhor BWT. A partir da Tabela 2a, vemos que, assim como no MNIST, a adição do discriminador melhorou a acurácia em 2 dos 3 experimentos realizados, dessa vez os casos onde o modelo foi treinado por 10 e 20 épocas foram melhores com o discriminador do que sem ele.

A Tabela 2b mostra que, no caso do BWT, a inclusão do discriminador produziu resultados melhores do que a arquitetura sem ele em todos os experimentos, superando inclusive a arquitetura original no treinamento com 5 e 10 épocas.

	5 épocas	10 épocas	20 épocas
IRCL Original	76,066%	**76,544%**	75,596%
Nosso (sem Disc.)	64,069%	56,841%	51,764%
Nosso (com Disc.)	60,988%	58,645%	56,266%

(a) Acurácia média

	5 épocas	10 épocas	20 épocas
IRCL Original	-20,252%	-20,612%	-22,504%
Nosso (sem Disc.)	-20,452%	-24,468%	-35,500%
Nosso (com Disc.)	**-17,301%**	-20,347%	-27,083%

(b) BWT médio

Tabela 2: Resultados obtidos no *Fashion*-MNIST

A Figura 13 contém as imagens geradas pelos modelos após o treinamento da última tarefa do *Fashion*-MNIST por 20 épocas. Assim como no caso do MNIST, as imagens geradas pelas nossas implementações possuem uma nitidez maior do que as que foram geradas pela arquitetura original. Também como ocorreu no MNIST, os nossos modelos geraram algumas imagens difíceis de interpretar, diferentemente das imagens produzidas pela implementação original do IRCL.

(a) IRCL Original (b) Nosso (sem Disc.) (c) Nosso (com Disc.)

Figura 13: Imagens do *Fashion*-MNIST geradas pelos modelos treinados por 20 épocas

5.2.3 EMNIST

Os resultados obtidos no EMNIST são apresentados na Tabela 3. É possível verificar a partir da Tabela 3a que a acurácia apresentada pelos modelos é consideravelmente pior do

que o que foi conseguido ao utilizar o MNIST e *Fashion*-MNIST, com uma diferença de pelo menos 20 pontos percentuais em relação ao que foi conseguido com esses *datasets*. Isso mostra a dificuldade que os modelos têm em lidar com *datasets* que possuem uma maior quantidade de classes.

Apesar disso, a implementação original do IRCL ainda apresentou resultados de acurácia melhores do que a arquitetura com as alterações que nós implementamos. Ao comparar nossa arquitetura com e sem o discriminador, vemos que o uso desse componente gerou melhores resultados quando o modelo foi treinado por 5 e 10 épocas. Ao avaliar o BWT a partir da Tabela 3b, vemos que a nossa arquitetura conseguiu resultados melhores do que a implementação original nos 3 casos avaliados.

	5 épocas	10 épocas	20 épocas
IRCL Original	**56,666%**	51,452%	45,874%
Nosso (s/ Disc.)	33,842%	34,136%	35,262%
Nosso (c/ Disc.)	39,548%	35,800%	33,683%

(a) Acurácia média

	5 épocas	10 épocas	20 épocas
IRCL Original	-40,225%	-46,750%	-53,100%
Nosso (sem Disc.)	-32,442%	-38,403%	**-32,142%**
Nosso (com Disc.)	-33,879%	-35,415%	-42,100%

(b) BWT médio

Tabela 3: Resultados obtidos no EMNIST

A Figura 14 ilustra a dificuldade dos modelos em lidar com conjuntos de dados com mais classes. As 3 arquiteturas geraram imagens com pouca definição e produziram imagens difíceis de reconhecer visualmente. Apesar disso, é possível identificar algumas imagens produzidas pelos nossos modelos que apresentam uma boa definição, principalmente as letras das últimas tarefas como o X, Y e Z nas Figuras 14b e 14c. No entanto, muitas das imagens produzidas não são possíveis de serem identificada. A implementação original, por sua vez, apresentou imagens com pouca definição para todas as classes do *dataset*, mas ainda é possível reconhecer a classe de uma quantidade maior de imagens.

(a) IRCL Original (b) Nosso (sem Disc.) (c) Nosso (com Disc.)

Figura 14: Imagens do EMNIST geradas pelos modelos treinados por 20 épocas

5.2.4 CIFAR-10

A Tabela 4 contém os resultados obtidos no *dataset* CIFAR-10. Como pode ser visto na Tabela 4a, nenhuma das arquiteturas foi capaz de apresentar uma acurácia acima de 20%. Isso demonstra a complexidade adicionada por *datasets* com imagens que possuem mais detalhadas.

Ao treinar a nossa arquitetura sem discriminador por 10 épocas, o resultado foi praticamente o mesmo do que seria conseguido ao fazer a classificação a partir de escolha aleatória, indicando que durante o treinamento houve um esquecimento quase que completo do que foi aprendido em cada tarefa. Apesar disso, ainda é possível observar que a presença do discriminador melhorou a acurácia nos 3 casos avaliados.

Os resultados ruins de acurácia refletem o que mostra a Tabela 4b, onde todos os experimentos apresentaram valores muito negativos de BWT, o que reforça que os modelos estão de fato sofrendo o esquecimento catastrófico. Ainda assim, podemos ver que, mais uma vez, a nossa arquitetura apresentou os melhores valores de BWT, dessa vez ao treinar o modelo por 10 e 20 épocas.

	5 épocas	10 épocas	20 épocas
IRCL Original	**17,990%**	17,578%	16,510%
Nosso (sem Disc.)	14,536%	10,689%	15,097%
Nosso (com Disc.)	14,780%	12,105%	16,298%

(a) Acurácia média

	5 épocas	10 épocas	20 épocas
IRCL Original	-47,252%	-52,326%	-59,696%
Nosso (sem Disc.)	-53,208%	-48,714%	-44,595%
Nosso (com Disc.)	-54,635%	**-42,613%**	-44,119%

(b) BWT médio

Tabela 4: Resultados obtidos no CIFAR-10

A Figura 15 apresenta as imagens geradas a partir do CIFAR-10. Aqui é possível visualizar uma das causas do baixo desempenho de classificação apresentado pelos modelos. Nenhuma das imagens geradas possui uma forma bem definida, além de não ser possível identificar visualmente as classes as quais elas pertencem. As imagens produzidas pelos nossos modelos e pela arquitetura original possuem características bem diferentes, enquanto os nossos modelos geraram imagens com mais características de alta frequência, as imagens geradas pelo modelo original possuem mais características de baixa frequência.

(a) IRCL Original (b) Nosso (sem Disc.) (c) Nosso (com Disc.)

Figura 15: Imagens do CIFAR-10 geradas pelos modelos treinados por 20 épocas

6

CONCLUSÃO

Neste trabalho, nós propusemos a utilização de camadas convolucionais e de um modelo generativo mais robusto na arquitetura proposta em IRCL. Utilizamos uma versão condicional do *autoencoder* LBAE como modelo generativo e adicionamos a ele um discriminador para avaliar as imagens produzidas, como é feito nas GANs. Além disso, reimplementamos o módulo específico da arquitetura do IRCL utilizando camadas convolucionais e fazendo com que a representação gerada estivesse no mesmo espaço de Bernoulli utilizado pelo LBAE.

A partir dos resultados obtidos, fica claro que as alterações propostas não são eficientes para melhorar o desempenho da classificação em relação ao que foi apresentado pela implementação original da arquitetura. Nossa implementação não foi capaz de superar a acurácia do modelo original em nenhum dos experimentos realizados. Isso pode ser explicado pelo fato de que, apesar de ser capaz de gerar imagens mais nítidas, o modelo gera mais imagens sem forma bem definida do que o VAE original.

No entanto, fica claro também que a adição de um discriminador ao nosso modelo para avaliar as imagens produzidas pelo *autoencoder* tem efeito positivo na acurácia quando comparado à arquitetura sem esse componente. A utilização do discriminador melhorou a acurácia em pelo menos 2 dos 3 experimentos nos 4 *datasets* avaliados.

Apesar de não melhorar a acurácia, nossas alterações causaram efeitos positivos no BWT. Nossa implementação foi capaz de obter resultados melhores do que a original em 3 dos 4 *datasets*. Isso indica que as alterações implementadas podem causar efeitos positivos, deixando espaço para que se investigue com mais profundidade cada uma delas para encontrar formas de melhorar o desempenho geral da tarefa de classificação.

6.1 Trabalhos Futuros

Como visto no Capítulo 5, o CLBAE gerou muitas imagens sem forma definida e difíceis de serem reconhecidas. Dessa forma, uma possibilidade para futuros trabalhos é avaliar a utilização de modelos generativos capazes de gerar imagens com maior qualidade, como por exemplo o NVAE ou VQ-VAE2. Além disso, como a utilização de um discriminador mostrou melhorar os resultados obtidos na classificação, outra possibilidade a ser explorada é a adição de outros componentes que se mostraram eficazes para melhorar imagens produzidas por GANs.

REFERÊNCIAS

[1] R. Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040, 2020.

[2] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C):109–165, January 1989.

[3] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[4] Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Learning invariant representation for continual learning. In *Meta-Learning for Computer Vision Workshop at the 35th AAAI Conference on Artificial Intelligence (AAAI-21)*, 2021.

[5] Jary Pomponi, Simone Scardapane, and Aurelio Uncini. Pseudo-rehearsal for continual learning with normalizing flows. *arXiv preprint arXiv:2007.02443*, 2020.

[6] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[7] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2994–3003, 2017.

[8] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[9] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In *NeurIPS Continual learning Workshop*, 2018.

[10] Achraf Oussidi and Azeddine Elhassouny. Deep generative models: Survey. In *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–8, 2018.

[11] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[13] Jiri Fajtl, Vasileios Argyriou, Dorothy Monekosso, and Paolo Remagnino. Latent bernoulli autoencoder. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2964–2974. PMLR, 13–18 Jul 2020.

[14] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[15] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQVAE-2. In *Advances in neural information processing systems*, pages 14866–14876, 2019.

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[18] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[21] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[22] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[23] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[24] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[25] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[26] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *Computer Vision – ECCV 2020*, pages 386–402, Cham, 2020. Springer International Publishing.

[27] Jiawei Chen, Janusz Konrad, and Prakash Ishwar. A cyclically-trained adversarial network for invariant representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3393–3402, 2020.

[28] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[30] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[31] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *CoRR*, abs/1702.08658, 2017.

[32] Lei Cai, Hongyang Gao, and Shuiwang Ji. Multi-stage variational auto-encoders for coarse-to-fine image generation. *CoRR*, abs/1705.07202, 2017.

[33] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1558–1566. JMLR.org, 2016.

[34] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6470–6479, 2017.