Estudo de Seleção de Instâncias em Aprendizado de Métrica para o problema de classificação com Aprendizado Incremental

Luiz Santos
Universidade Federal de Mato Grosso do Sul
Campo Grande, Brasil
luiz.h.s.santos@ufms.br

Matheus Vyctor Espíndola
Universidade Federal de Mato Grosso do Sul
Campo Grande, Brasil
matheus_vyctor@ufms.br

Edson Takashi Matsubara
Universidade Federal de Mato Grosso do Sul
Campo Grande, Brasil
edson.matsubara@ufms.br

***Resumo*—Aprendizado Incremental lida com um aumento contínuo no número de classes ao longo do tempo. O principal problema nesse contexto é o "esquecimento catastrófico", em que os modelos perdem a capacidade de resolver as tarefas originalmente aprendidas ao serem treinados para novas tarefas; além da necessidade de cada vez mais recursos computacionais conforme novos treinos acontençam. Para contornar esse problema a proposta avalia o uso de modelos de Aprendizado Profundo de Métricas juntamente com técnicas de Seleção de Instâncias no conjunto de dados CIFAR-100. Na valiação experimental os *embeddings* foram avaliados com *K-Nearest Neighbor* para análise de desempenho. O modelo *EfficientNetV2 Small* foi escolhido como *backbone*. Para gerar representações numéricas de alta dimensionalidade que refletem a similaridade entre as classes foi treinado um modelo de Aprendizado Profundo de Métricas, chamado *Zero-shot*. Em seguida, o *fine-tuning* do modelo *Zero-shot* foi realizado para aprender novas classes adicionadas ao conjunto de dados original. Para garantir que o modelo não esquecesse as classes antigas, técnicas de Seleção de Instâncias foram utilizadas, como o Random Mutation Hill Climbing (RMHC) e o Gaussian Mixture (GM). O intuito é selecionar instâncias representativas das classes antigas e reduzir ruídos. O resultado obtido obteve uma acurácia de $78,80\%$ para o conjunto de dados original (Tarefa 1), e uma acurácia de $82,00\%$ para o conjunto com novas classes (Tarefa 2).Comparado com o modelo *Baseline*, houve uma redução de $9,33\%$ para a Tarefa 1 – a qual o modelo Proposto foi inicialmente treinado – e um aumento em $2,12\%$ de acurácia para a Tarefa 2. Já a quantidade de instâncias necessárias para o treino do modelo Proposto foi $81,00\%$ menor do que o modelo *Baseline*, e a quantidade de épocas foi $87,50\%$ menor. Dessa forma, o estudo propõem um treino rápido com tamanho limitado do conjunto de treino que evita o "esquecimento catastrófico"das classes antigas.**

***Palavras-chave*—Aprendizado de Métrica, Seleção de Instâncias, Aprendizado de Máquina, Aprendizado Incremental, Aprendizado Profundo.**

I. Introdução

Identificação de espécies de animais e reconhecimento facial são exemplos de problemas reais onde o número de classes (espécies/pessoas) facilmente ultrapassa algumas centenas ou milhares. O desenvolvimento de modelos para estes cenários precisa lidar com dois problemas: a inclusão de novas classes e o grande volume de dados para treinamento. A inclusão de novas classes é um problema conhecido na literatura como Aprendizado Incremental [5]. Um dos problemas de Aprendizado Incremental é o "esquecimento catastrófico" [4] que é caracterizado quando modelos "esquecem"a capacidade de resolver a tarefa a qual foram treinados originalmente após serem retreinados para uma nova tarefa. Isso ocorre porque o conhecimento atual pode ser sobreescrito ou distorcido ao alterar os parâmetros do modelo para acomodar novas classes.

A solução clássica para esse problema seria treinar um modelo de classificação cada vez que novas classes são inseridas [21], porém isso não permite escalar de forma eficiente. Uma maneira de contornar essa situação é usando algoritmos que não precisam aprender novos parâmetros para novas classes e mantenham o conhecimento adquirido.

Aprendizado de Métrica é uma possível solução a este problema pois consiste em uma abordagem de aprendizado de máquina que visa definir similaridades e dissimilaridades entre objetos [6]. Essa técnica permite que os dados sejam representados por pontos em um espaço métrico em comum de maneira que eles possam ser agrupados de acordo com suas características e relações semelhantes – tendo a capacidade de extrair características para classes que não foram previamente encontradas durante o treinamento – o que facilita a identificação de padrões e a tomada de decisões baseadas na proximidade dos objetos no espaço de representação. Ou seja, a representação é feita de tal maneira que os *embeddings* de duas imagens sejam mais próximos a medida que as imagens possuam mais características similares. Tornando-se, assim, útil na solução de problemas de visão computacional, tais como reconhecimento facial [16] e *clustering* [13].

Um benefício da abordagem utilizando Aprendizado de Métrica é que ela é concebida para lidar com a tarefa de *zero-shot learning*, onde é necessário extrair características de classes nunca antes vistas por criar um espaço métrico comum. Isso significa que o modelo aprende a generalizar e pode realizar inferências com base nas similaridades entre os objetos aprendidos. Essa capacidade de extrair características para classes não vistas anteriormente durante o treinamento é particularmente útil em cenários onde novas classes ou

exemplos continuamente surgem. Com o Aprendizado de Métrica, o modelo pode usar as características aprendidas a partir das classes conhecidas para inferir as características dos novos objetos e classificá-los adequadamente, mesmo sem ter visto exemplos específicos dessas classes anteriormente. Além disso, a criação do espaço métrico em comum possibilita o uso de técnicas de Seleção de Instâncias baseada em vetor de características, ou *embeddings*.

Uma das dificuldades para o uso de aprendizado de métrica é a inclusão de novas classes muito diferentes ao domínio original. Novas classes geram pontos no espaço aprendido que podem ser usados para estimar similaridade; entretanto, como o algoritmo não aprendeu a discernir as novas classes entre si e entre as restantes, a região predita no espaço pode levar a resultados errôneos [9]. O modelo deve ser retreinado com todos os dados para garantir consistência e evitar resultados indesejados, levando a um conjunto de treino crescente que necessita de cada vez mais recursos computacionais. Esse processo contínuo de retreinamento se torna oneroso, tornando-se impraticável em cenários onde há um número de classes em constante evolução.

Dessa forma, é necessário adotar estratégias que reduzam ruídos e *outliers* das classes antigas, a fim de garantir que o modelo seja capaz de aprender novas classes sem esquecer das classes antigas, também reduzindo a quantidade de instâncias necessárias para o treino. Para isso, a Seleção de Instâncias é uma técnica estabelecida para resolver esse problema e possui algoritmos bem conhecidos, como *Random Mutation Hill Climbing* (RMHC) [17] e *Gaussian Mixture* (GM) [15].

O objetivo do presente trabalho é unir Aprendizado Profundo de Métrica e Seleção de Instância em um problema de classificação com Aprendizado Incremental. A junção destas áreas possibilita realizar Aprendizado Incremental com menor esforço computacional obtido pela redução de instâncias de treinamento. As principais contribuições desse estudo são:

- Avaliar se uma abordagem de Aprendizado Incremental com o uso da técnica de Seleção de Instâncias é eficaz para resolver problemas de classificação com inserções de novas classes.
- Comparar diferentes algoritmos de Seleção de Instância no *fine tuning* para novas classes com a escolha aleatória dessas instâncias;
- Avaliar se uma abordagem de Aprendizado de Métricas com *zero-shot learning* é eficaz para resolver problemas de classificação com Aprendizado Incremental.
- A metodologia implementada traz uma nova abordagem para a literatura sobre o problema estudado, ao juntar técnicas de Seleção de Instâncias, Aprendizado Profundo de Métrica e Classificação com Aprendizado Incremental.

O restante desse artigo está dividido em: Materiais e métodos, onde será mostrado como foi a configuração dos experimentos realizados e as métricas utilizadas; O fluxo de trabalho proposto será debatido em Proposta; Resultados, em que será discutido os valores vistos nos testes, bem como serão comparados os diferentes modelos experimentais; e Conclusão, para sumarizar os pontos importantes e debater futuros passos.

II. Materiais e métodos

A. Aprendizado Profundo de Métricas

A função de custo usada para o Aprendizado de Métrica foi a SubCenterArcFaceLoss [2], implementada pela biblioteca Pytorch Metric Learning [12]. Essa escolha de loss visou uma abordagem que pudesse melhorar a robustez do modelo em ambientes não controlados, principalmente em relação ao ruído presente nas amostras de treinamento.

A *SubCenterArcFaceLoss* relaxa as restrições intraclasse do método ArcFace [3], permitindo a existência de vários subcentros para cada classe. Assim, as amostras de treinamento precisam estar próximas a qualquer um dos subcentros positivos, em vez de apenas um centro positivo. Essa técnica promove um subgrupo dominante contendo a maioria das amostras limpas da classe e subgrupos não dominantes que podem conter amostras difíceis ou com ruído. Após o modelo adquirir poder discriminativo suficiente, subcentros não dominantes e amostras ruidosas de alta confiança podem ser descartadas. Essa abordagem ajuda a restaurar a coesão intraclasse, reduz o impacto do ruído e obtém desempenho comparável ao ArcFace treinado com conjuntos de dados limpos manualmente [2].

Já o otimizador usado, foi o *Adam* [7]. *Adam* é um algoritmo estocástico de otimização de funções objetivas baseado em gradientes de primeira ordem. Ele se destaca por suas vantagens, como estimativa adaptativa de momentos de baixa ordem, simplicidade de implementação, alta eficiência computacional, baixo consumo de memória e invariância relacionada ao redimensionamento diagonal do gradiente [7]; sendo estes os motivos de sua escolha no presente estudo. Além disso, o Adam é útil não só em problemas com grandes quantidades de dados e/ou parâmetros, objetivos não estacionários e gradientes ruidosos e/ou esparsos; como também possui hiperparâmetros que geralmente requerem poucos ajustes [7].

Os modelos foram treinados no EfficientNetV2 Small [18] pré-treinado no ImageNet [1] obtido do repositório Pytorch Image Models [20]. EfficientNetV2 é uma família de arquiteturas de redes neurais convolucionais que são computacionalmente eficientes em tarefas de visão computacional [18]. O modelo EfficientNetV2 Small foi escolhido para alcançar eficiência e desempenho sem afetar significativamente as métricas de avaliação . Essa escolha é ditada pela capacidade do modelo de equilibrar a eficiência computacional com resultados de alto nível.

B. Seleção de Instâncias

Os experimentos foram avaliados em dois algotimos de seleção de instância: *Random Mutation Hill Climbing* (RMHC) e *Gaussian Mixture* (GM).

RMHC é baseado em busca local e consiste em dividir de forma aleatória o conjunto de treino T para um conjunto de selecionados S. A cada iteração do algoritmo, um elemento de S é trocado aleatoriamente por um elemento de T. Se a função de ajuste para o novo conjunto selecionado for

melhor que para o conjunto antigo, então a troca se mantém, senão, ela é desfeita. Após várias iterações, a função retorna o melhor conjunto S encontrado. No presente estudo, foi adotada a acurácia de um classificador KNN treinado com o novo conjunto S como função de ajuste, seguindo o que é comumente utilizado na literatura [17].

GM é um modelo probabilístico que assume que todos os dados são gerados pela mistura de um finito número de distribuições gaussianas com parâmetros desconhecidos. Esse modelo é usado na seleção de instâncias para computar a função de probabilidade para cada amostra. A seleção é feita com base nas instâncias de maior valor. Essa abordagem probabilística do GM permite selecionar um conjunto representativo das classes antigas, considerando as características e a distribuição dos dados.

C. Métricas

Os experimentos foram avaliados usando 4 métricas: Acurácia, Precisão, *Recall* e *F1-score*. Todos as métricas mostradas foram calculadas com média macro. Considere C o número total de classes e N o número total de amostras.

Acurácia: A acurácia é uma medida comum de desempenho para problemas de classificação multiclasse. Ela calcula a proporção de amostras classificadas corretamente em relação ao total de amostras. A fórmula da acurácia é:

$$\text{Acurácia} = \frac{\sum_{i=1}^{C} \text{VP}_i + \text{VN}_i}{N}$$

Onde VP_i são os verdadeiros positivos da classe i e VN_i são os verdadeiros negativos da classe i.

Precisão: A precisão mede a proporção de amostras classificadas como positivas que são realmente positivas. O cálculo na média macro é feita para cada classe individualmente e tirando a média dessas precisões. A fórmula da precisão com média macro é:

$$\text{Precisão} = \frac{1}{C} \sum_{i=1}^{C} \frac{\text{VP}_i}{\text{VP}_i + \text{FP}_i}$$

Onde VP_i são os verdadeiros positivos da classe i e FP_i são os falsos positivos da classe i.

Recall: O recall, também conhecido como taxa de verdadeiros positivos, mede a proporção de amostras positivas corretamente classificadas em relação ao total de amostras positivas. Em outras palavras, é a capacidade do modelo de encontrar todas as amostras positivas. Para múltiplas classes, o valor final é obtido tirando a média dos recalls de cada classe. A fórmula do recall com média macro é:

$$\text{Recall} = \frac{1}{C} \sum_{i=1}^{C} \frac{\text{VP}_i}{\text{VP}_i + \text{FN}_i}$$

Onde VP_i são os verdadeiros positivos da classe i e FN_i são os falsos negativos da classe i.

F1-score: A pontuação F1 para problemas de classificação multiclasse com média macro é calculada levando em consideração a precisão e o recall para cada classe individualmente e, em seguida, tirando a média das pontuações F1 dessas classes. A fórmula do F1-Score com média macro é:

$$\text{F1-Score} = \frac{1}{C} \sum_{i=1}^{C} \frac{2 \times \text{Precisão}_i \times \text{Recall}_i}{\text{Precisão}_i + \text{Recall}_i}$$

As métricas foram calculadas utilizando a biblioteca *python* Scikit-learn [14].

III. Proposta

A. Definição

A Figura 1 mostra o fluxograma proposto. Dois problems são definidos:

(1) **Problema Original (Tarefa 1)**: aprender a classificar o conjunto de dados original.
(2) **Problema Novo (Tarefa 2)**: aprender a classificar o conjunto de dados com novas classes.

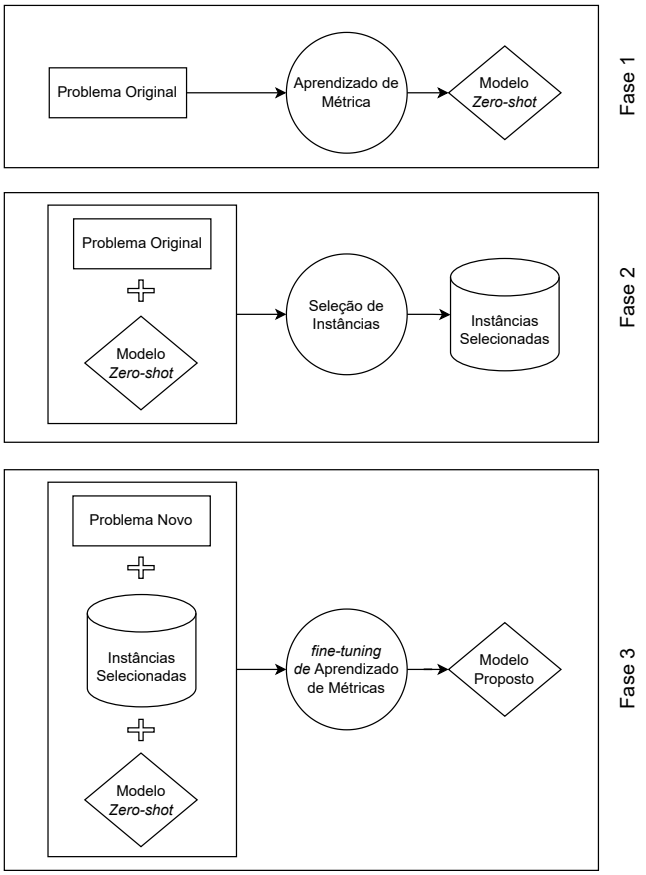

Figura 1. Fluxograma do processo proposto.

O fluxograma proposto é composto por 3 fases, sendo elas:

- **Fase 1**: a primeira fase tem como objetivo a criação de um modelo de Aprendizado de Métrica para o Problema Original (Tarefa 1). Para isso, um modelo chamado de *Zero-shot* é treinado com o conjunto original de treino e validação.

- **Fase 2**: a segunda fase tem como objetivo selecionar instâncias do Problema Original (1). Para isso, a técnica de Seleção de Instância é utilizada com os *embeddings* gerados pelo modelo *Zero-shot* do conjunto de dados do Problema Original.
- **Fase 3**: a terceira fase tem como objetivo a criação de um modelo de Aprendizado de Métrica para o Problema Novo (Tarefa 2) mantendo o que foi aprendido para o Problema Original (Tarefa 1). Para isso, um modelo chamado de *Proposto* passa por um *fine-tuning* de Aprendizado de Métrica do modelo *Zero-shot* com as instâncias selecionadas do Problema Original (Tarefa 1) concatenadas com o conjunto de dados do Problema Novo (Tarefa 2).

A abordagem proposta foi comparada com outros 3 modelos:

- ***Baseline***: um modelo treinado com todas as 100 classes.
- ***Zero-shot***: modelo obtido pela Fase 1.
- ***Random***: modelo obtido ao trocar a técnica de Seleção de Instância na Fase 2 por uma seleção aleatória.

B. Experimentação do modelo Baseline

O modelo *Baseline* foi treinado com todas as classes do conjunto de dados (dados do Problema Original mais os dados do Problema Novo) conforme a Figura 2.

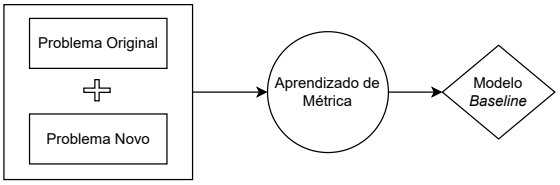

Figura 2. Fluxograma do Baseline.

C. Experimentação

CIFAR-100 [8] foi a base de dados utilizada para a presente pesquisa. Ele possui 60.000 imagens divididas em 100 classes distintas. As classes são uniformemente distribuídas, tendo cada uma, assim, 600 amostras representativas; as quais podem ser agrupadas em 20 superclasses, conforme mostra a Tabela I.

O conjunto de dados do CIFAR-100 foi dividido conforme a Figura 3. Para o modelo *Baseline* a divisão foi 80% treino (48.000 imagens), 10% validação (6.000 imagens) e 10% teste (6.000 imagens).

Para a Tarefa de Aprendizado Incremental (problema novo), as 10 últimas classes do CIFAR-100 – que remetem a veículos – foram separadas. A seleção dessa categoria foi feita por ser a mais diferente e difícil de predizer comparada às outras classes do conjunto. As classes restantes compõem a Tarefa 1 (problema original). Cada classe foi dividida em 80% treino, 10% validação e 10% teste. A tabela II mostra distribuição do conjunto em treino, validação e teste para cada Tarefa.

TABELA I
CIFAR-100 SUPER CLASSES E CLASSES

Super Classes	**Classes**
mamíferos aquáticos	castor, golfinho, lontra, foca, baleia
peixes	peixes de aquário, linguado, raia, tubarão, truta
flores	orquídeas, papoulas, rosas, girassóis, tulipas
recipientes de comida	garrafas, tigelas, latas, copos, pratos
frutas e legumes	maçãs, cogumelos, laranjas, peras, pimentões
dispositivos elétricos domésticos	relógio, teclado de computador, lâmpada, telefone, televisão
móveis domésticos	cama, cadeira, sofá, mesa, guarda-roupa
insetos	abelha, besouro, borboleta, lagarta, barata
grandes carnívoros	urso, leopardo, leão, tigre, lobo
grandes construções ao ar livre	ponte, castelo, casa, estrada, arranha-céu
grandes paisagens naturais ao ar livre	nuvem, floresta, montanha, planície, mar
grandes onívoros e herbívoros	camelo, gado, chimpanzé, elefante, canguru
mamíferos de tamanho médio	raposa, ouriço, gambá, guaxinim, doninha
invertebrados não insetos	caranguejo, lagosta, caracol, aranha, minhoca
pessoas	bebê, menino, menina, homem, mulher
répteis	crocodilo, dinossauro, lagarto, cobra, tartaruga
pequenos mamíferos	hamster, rato, coelho, musaranho, esquilo
árvores	bordo, carvalho, palmeira, pinheiro, salgueiro
veículos 1	bicicleta, ônibus, motocicleta, caminhonete, trem
veículos 2	cortador de grama, foguete, bonde, tanque, trator

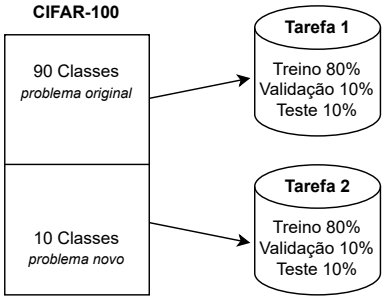

Figura 3. Divisão do CIFAR-100 para cada problema.

Para todos os experimentos, as imagens foram redimensionadas para 224x224, viradas horizontalmente com uma probabilidade de 50% e aplicadas *TrivialAugment* [11].

A política de treinamento da Fase 1 foi usar as 7 épocas iniciais para *warmup* fazendo um aumento linear da taxa

TABELA II
DIVISÃO DO CONJUNTO DE DADOS PARA CADA TAREFA

Tarefa	**Classes**	**Treino**	**Validação**	**Teste**	**Treino e Validação**
1	90	43.200	5.400	5.400	48.600
2	10	4.800	600	600	5.400

de aprendizado até um valor máximo de $3 \cdot 10^{-4}$. *Cosine Annealing* [10] foi feito para as 33 épocas restantes.

Na Fase 2, a taxa de retenção dos algoritmos de Seleção de Instância foi de 10%. O método RMHC realizou 5000 iterações.

Na Fase 3, 5 épocas foram usadas sob a política tradicional de treinamento para o *fine-tuning*.

Para A classificação dos pontos para as classes é feita com KNN usando a implementação do Scikit-learn [14]. Todos os modelos foram treinados com o valor de $k = 3$. A implementação do código foi baseada nas boas-práticas de estado-da-arte conforme [19].

IV. Resultados

A. Resultados das Experimentações

Esta seção apresenta os resultados obtidos após experimentações e avaliações das abordagens utilizadas nos modelos de Aprendizado Profundo de Métricas. Uma análise comparativa foi realizada para investigar o comportamento do Aprendizado Incremental ao adicionar novas classes e avaliar se houve melhorias ao utilizar a técnica de Seleção de Instâncias. A Tabela III fornece uma visão geral dos resultados obtidos por cada modelo em termos de acurácia, precisão, *recall* e *F1-score*.

No contexto do Aprendizado Incremental abordado, as classes selecionadas para o treinamento foram as últimas 10, referentes a veículos. Essas classes não possuíam semelhanças com as outras 90 classes, o que exigiu que o modelo aprendesse suas características quase do zero. No entanto, cenários de Aprendizado Incremental em que as novas classes possuem características semelhantes às classes existentes são comuns, facilitando o ajuste fino. O cenário proposto neste estudo visou o pior caso de Aprendizado Incremental.

B. Modelo Baseline

A primeira análise foi verificar como um modelo treinado com todo o conjunto de dados do CIFAR-100, sem a divisão entre as Tarefas 1 e 2, se comportaria. O maior problema dessa abordagem é que ela não faz o uso de Aprendizado Incremental, sendo necessário um novo treino sempre que houver um conjunto de novas classes a serem adicionadas. Além disso, como não é utilizada a técnica de Seleção de Instâncias, todo novo treinamento será realizado com todo o *dataset* já treinado anteriormente unido com as novas classes a serem inseridas, fazendo-o com que o custo computacional se torne cada vez mais caro. Em contra-partida, esse foi o modelo que obteve as melhores métricas quando analisadas com as Tarefas 1 e 2, com $0,863$ de acurácia, $0,865$ de precisão, $0,863$ de *recall* e $0,863$ de *F1-score*. Esse modelo será o *benchmark* de métricas escolhido para as demais análises.

Além disso, o *benchmark* estabelecido também apresentou resultados sólidos na comparação das Tarefas 1 e 2. Na Tarefa 1, alcançou uma acurácia de $0,869$ e um *F1-score* de $0,874$. Na Tarefa 2, obteve uma acurácia de $0,803$ e um *F1-score* de $0,886$.

C. Modelo Zero-shot

Esse modelo possui as melhores métricas para a Tarefa 1, com $0,904$ de acurácia e $0,904$ de *F1-score*. Isso vem do fato de que ele possui a mesma abordagem do *Baseline*, no entanto somente com 90 classes para serem aprendidas em vez de 100. No entanto, este modelo na Tarefa 2, que possuíam 10 classes que ele nunca havia sido treinado, obteve o pior desempenho, com $0,580$ de acurácia e $0,712$ de *F1-score*. Embora seja concebido como um modelo de aprendizado profundo de métricas para *zero-shot learning*, isto é, ele consegue generalizar a extração de características para classes não vistas durante o treinamento, ele apresentou uma queda de 35,34% nessa Tarefa específica; em contraste do modelo com Aprendizado Incremental, que obteve métricas melhores nessa Tarefa e será discutido a seguir. As novas classes, responsáveis pela queda nas métricas do modelo *Zero-shot*, representam 10% do total da amostra para teste.

D. Modelo com Aprendizado Incremental e Seleção de Instâncias

Os resultados obtidos pelo modelo que utiliza Aprendizado Incremental com o método RMHC de Seleção de Instâncias com um taxa de retenção de 10% na Tarefa 1 foram $0,788$ de acurácia e $0,848$ de *F1-score*. Já com *Gaussian Mixture* e a mesma taxa de retenção as métricas subiram para $0,807$ de acurácia e $0,866$ de *F1-score*.

Já referente aos resultados para a Tarefa 2, o método RMHC obteve $0,820$ de acurácia e $0,868$ de *F1-score*. Já o método GM obteve $0,785$ de acurácia e $0,842$ de *F1-score*, sendo este o que possui o melhor resultado das 4 métricas em conjunto para a Tarefa 2 dentre todas as abordagens analisadas. Esses resultados demonstram que ao adotar a estratégia de Aprendizado Incremental, esses modelos foram capazes de lidar de forma eficiente com a adição de novas classes, obtendo um desempenho superior em comparação com outras abordagens para a Tarefa 2.

Assim sendo, ao juntar as Tarefas 1 e 2, como ambos os modelos tiveram resultados significativos para ambas as tarefas, o resultado final acompanhará essa distribuição. Portanto, o método RMHC obteve $0,791$ de acurácia e $0,808$ de *F1-score*. Já o modelo com GM obteve $0,805$ de acurácia e $0,823$ de *F1-score*, levemente superior comparado com o RMHC.

Ambas as técnicas de Seleção de Instâncias alcançaram altas métricas de precisão nos experimentos: O método RMHC obteve uma precisão de $0,948$ na Tarefa 1, $0,928$ na Tarefa 2 e $0,872$ em ambas. Já o método GM alcançou uma precisão de $0,952$ na Tarefa 1, $0,912$ na Tarefa 2 e $0,875$ em ambas. Esses valores são muito próximos, em alguns casos maiores, aos resultados do próprio *benchmark* estabelecido. No entanto, o *recall* deles ficaram ligeiramente abaixo, com o RMHC apresentando $0,788$, $0,820$ e $0,791$ nas Tarefas 1, 2 e ambas, respectivamente, e o GM com $0,807$, $0,785$ e $0,805$, respectivamente.

A alta precisão indica que o modelo tem poucos falsos positivos, ou seja, poucas instâncias negativas são erroneamente classificadas como positivas. No entanto, um baixo

TABELA III
COMPARAÇÃO DOS RESULTADOS COM MÉDIA MACRO PARA CADA ABORDAGEM REALIZADA.

Modelo	Conjunto treino (KNN)	Conjunto teste (KNN)	Acurácia	Precisão	*Recall*	*F1-score*
Baseline	T1+T2	T1	0,869	0,882	0,869	0,874
Zero-shot	T1	T1	**0,904**	0,906	**0,904**	**0,904**
Random	IR(T1)+T2	T1	0,779	0,944	0,779	0,839
Proposta com RMHC 10%	IS(T1)+T2	T1	0,788	0,948	0,788	0,848
***Proposta com RMHC 10%**	T1+T2	T1	0,865	0,923	0,865	0,890
Proposta com GM 10%	IS(T1)+T2	T1	0,807	**0,952**	0,807	0,866
***Proposta com GM 10%**	T1+T2	T1	0,863	0,924	0,863	0,890
Baseline	T1+T2	T2	0,803	**0,998**	0,803	**0,886**
Zero-shot	T1+T2	T2	0,580	0,956	0,580	0,712
Random	IR(T1)+T2	T2	**0,823**	0,919	**0,823**	0,867
Proposta com RMHC 10%	IS(T1)+T2	T2	0,820	0,928	0,820	0,868
***Proposta com RMHC 10%**	T1+T2	T2	0,652	0,944	0,652	0,763
Proposta com GM 10%	IS(T1)+T2	T2	0,785	0,912	0,785	0,842
***Proposta com GM 10%**	T1+T2	T2	0,648	0,941	0,648	0,761
Baseline	T1+T2	T1+T2	**0,863**	0,865	**0,863**	**0,863**
Zero-shot	T1+T2	T1+T2	0,853	0,853	0,853	0,851
Random	IR(T1)+T2	T1+T2	0,783	0,870	0,783	0,799
Proposta com RMHC 10%	IS(T1)+T2	T1+T2	0,791	0,872	0,791	0,808
***Proposta com RMHC 10%**	T1+T2	T1+T2	0,844	0,853	0,844	0,846
Proposta com GM 10%	IS(T1)+T2	T1+T2	0,805	**0,875**	0,805	0,823
***Proposta com GM 10%**	T1+T2	T1+T2	0,841	0,853	0,841	0,845

* Apesar do Proposto ter sido treinado com as instâncias selecionadas, a KNN desse experimento foi treinada com todo o conjunto (T1+T2).
(IS) - Instâncias selecionadas; (IR) - Instâncias selecionadas randomicamente;

TABELA IV
COMPARAÇÃO DO TAMANHO DO CONJUNTO DE DADOS DE TREINO COM A ACURÁCIA OBTIDA PARA CADA ABORDAGEM REALIZADA.

Modelo	Quantidade de instâncias no treino	Quantidade de épocas de treino	Conjunto teste	Acurácia
Baseline	54.000	40	T1	0,869
Zero-shot	48.600	40	T1	**0,904**
Proposta com RMHC 10%	10.260	5	T1	0,788
Proposta com GM 10%	10.260	5	T1	0,807
Baseline	54.000	40	T2	0,803
Zero-shot	54.000	40	T2	0,580
Proposta com RMHC 10%	10.260	5	T2	**0,820**
Proposta com GM 10%	10.260	5	T2	0,785
Baseline	54.000	40	T1+T2	**0,863**
Zero-shot	54.000	40	T1+T2	0,853
Proposta com RMHC 10%	10.260	5	T1+T2	0,791
Proposta com GM 10%	10.260	5	T1+T2	0,805

recall sugere que ele pode estar perdendo muitas instâncias positivas, classificando-as como negativas. Alcançar tanto alta precisão quanto alto *recall* é desejável, mas há um *trade-off* entre essas métricas, ou seja, aumentar uma pode resultar na diminuição da outra. A escolha entre maximizar a precisão ou o *recall* depende do contexto e das necessidades do problema em questão. Essa diferença destaca a necessidade de considerar tanto a precisão quanto o *recall* ao avaliar o desempenho geral dos modelos.

E. Seleção de Instâncias Randômica (IR)

Durante os experimentos, uma análise para validar a relevância da técnica de Seleção de Instâncias foi feita nas resoluções das tarefas propostas. Para isso, as duas abordagens – o método RMHC e o método GM – foram comparadas com uma terceira abordagem que selecionava uma subamostra de forma randômica, mantendo a taxa de retenção de 10%. Os resultados obtidos pelo modelo *Random* foram muito próximos, porém inferiores aos métodos RMHC e GM em ambas as tarefas.

F. KNN treinado sem Seleção de Instâncias

Durante a experimentação, uma nova abordagem que envolveu o treinamento da KNN com o conjunto de dados completo das Tarefas 1 e 2 combinadas foi avaliada. Assim, o KNN recebe como entrada os *embeddings* gerados pelo modelo Proposto, porém eles são referentes a todo o conjunto de treino e validação, sem uma seleção inteligente de amostras.

Essa abordagem resultou em maiores acurácias para a Tarefa 1 em ambos os métodos, RMHC e GM. Para o RMHC, a acurácia aumentou de $0,788$ para $0,865$, enquanto para o GM, a acurácia subiu de $0,807$ para $0,863$. No entanto, na Tarefa 2, essa abordagem teve um efeito negativo, resultando em uma queda na acurácia. Para o RMHC, a acurácia diminuiu de $0,820$ para $0,652$, e para o GM, a acurácia caiu de $0,785$ para $0,648$. Esses resultados demonstram que, no experimento realizado, treinar o KNN com todo o conjunto de dados teve

um impacto positivo na acurácia da Tarefa 1, mas prejudicou na acurácia da Tarefa 2.

G. Comparação dos resultados de acurácia com o tamanho de conjunto de treino

A abordagem proposta – utilizando técnicas de Seleção de Instâncias (RMHC) e Aprendizado Incremental – obteve uma acurácia de $78,80\%$ para o conjunto de dados original (Tarefa 1), e uma acurácia de $82,00\%$ para o conjunto com novas classes (Tarefa 2) conforme a Tabela IV.

Comparando com o modelo *Zero-shot*, a acurácia para a Tarefa 2 aumentou em $38,45\%$, enquanto reduziu em $12,83\%$ para a Tarefa 1 – a qual foi inicialmente treinada.

Comparado com o modelo *Baseline*, a acurácia para a Tarefa 2 aumentou em $2,12\%$, enquanto reduziu em $9,33\%$ para a Tarefa 1. Já a quantidade de instâncias necessárias para o treino do modelo Proposto foi $81,00\%$ menor do que o modelo *Baseline*, e a quantidade de épocas foi $87,50\%$ menor.

V. Conclusão

Este estudo mostrou que a utilização de técnicas de Aprendizado Profundo de Métricas em conjunto com estratégias de Seleção de Instâncias pode ser uma abordagem promissora para o Aprendizado Incremental.

Os resultados mostraram que o *Baseline* treinado com todo o conjunto de dados teve resultados sólidos, porém com um alto custo computacional devido à necessidade de retreino com todos os dados cada nova inserção de classe. Por outro lado, a abordagem com o *zero-shot* evidenciou que o fato dele extrair características de classses nunca antes vistas, apesar de ser um modelo de *zero-shot learning*, gerou uma dificuldade significativa na Tarefa de aprendizado incremental, pois ele não aprende a refletir suas similaridades no espaço de dados. Além disso, o modelo *Random* mostrou que uma seleção inteligente de instância é importante para obtenção de melhores resultados.

Os resultados alcançados pela abordagem com aprendizado incremental e seleção de instâncias foram próximos ao *benchmark* estabelecido por (*Baseline*) para a Tarefa 2. Isso destaca a efetividade dessa abordagem na melhoria dos resultados, demonstrando sua capacidade de lidar com a adição de novas classes de forma adaptável e eficiente. Assim, a presente proposta se sobressai, pois possui um treino eficiente com resultados relevantes.

Para futuro trabalhos, pretende-se realizar mais testes com diferentes taxas de retenção, métodos de seleção de instância e outros conjuntos de dados, como o iNaturalist e Cars196. Outro ponto é testar esta proposta para mais rodadas de incremento dos dados e analisar o seu comportamento.

Agradecimentos

Gostaríamos de expressar nossa gratidão à Universidade Federal de Mato Grosso do Sul (UFMS), à Faculdade de Computação (FACOM) e ao Laboratório de Inteligência Artificial (LIA) pelo apoio concedido durante o desenvolvimento deste trabalho.

Também queremos estender nosso agradecimento a todos os nossos amigos, colegas e familiares que estiveram ao nosso lado ao longo dessa jornada, oferecendo suporte e encorajamento.

Por fim, gostaríamos de reconhecer a contribuição valiosa de todos os professores que nos acompanharam ao longo desses anos, compartilhando seus conhecimentos e orientações fundamentais para o nosso crescimento acadêmico.

Referências

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou. Sub-center arcface: Boosting face recognition by large-scale noisy web faces. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 741–757, Cham, 2020. Springer International Publishing.

[3] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4685–4694, 2019.

[4] Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[5] Bingchen Huang, Zhineng Chen, Peng Zhou, Jiayin Chen, and Zuxuan Wu. Resolving task confusion in dynamic expansion architectures for class incremental learning, 2023.

[6] Mahmut KAYA and Hasan Şakir BİLGE. Deep metric learning: A survey. *Symmetry*, 11(9), 2019.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[9] Xiaoxu Li, Xiaochen Yang, Zhanyu Ma, and Jing-Hao Xue. Deep metric learning for few-shot image classification: A review of recent developments, 2022.

[10] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016.

[11] Samuel G. Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 754–762, 2021.

[12] Kevin Musgrave, Serge J. Belongie, and Ser-Nam Lim. Pytorch metric learning. *ArXiv*, abs/2008.09164, 2020.

[13] Binh X. Nguyen, Binh D. Nguyen, Gustavo Carneiro, Erman Tjiputra, Quang D. Tran, and Thanh-Toan Do. Deep metric learning meets deep clustering: An novel unsupervised approach for feature embedding. *CoRR*, abs/2009.04091, 2020.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Douglas A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, 2009.

[16] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[17] David B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *International Conference on Machine Learning*, 1994.

[18] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139

of *Proceedings of Machine Learning Research*, pages 10096–10106. PMLR, 18–24 Jul 2021.

[19] Vasilis Vryniotis. How to train state-of-the-art models using torchvision's latest primitives, Nov 2021.

[20] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[21] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Deep class-incremental learning: A survey, 2023.