Exploração de estratégias para a classificação de fluxos de dados de imagens

Mateus Curcino de Lima

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2023

Mateus Curcino de Lima

Exploração de estratégias para a classificação de fluxos de dados de imagens

Tese de doutorado apresentada ao Programa de Pós-Graduaçãoda Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientadora: Prof.ª Dr.ª Maria Camila Nardini Barioni
Coorientadora: Prof.ª Dr.ª Elaine Ribeiro de Faria Paiva

Uberlândia
2023

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

L732 Lima, Mateus Curcino de, 1992-
2023 Exploração de estratégias para a classificação de fluxos de dados de imagens [recurso eletrônico] : Exploração de estratégias para a classificação de fluxos de dados de imagens / Mateus Curcino de Lima. - 2023.

Orientadora: Maria Camila Nardini Barioni.
Coorientadora: Elaine Ribeiro de Faria Paiva.
Tese (Doutorado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.te.2023.608
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Barioni, Maria Camila Nardini,1978-, (Orient.). II. Paiva, Elaine Ribeiro de Faria,1980-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Tese, 23/2023, PPGCO				
Data:	07 de novembro de 2023	Hora de início:	13:20	Hora de encerramento:	16:25
Matrícula do Discente:	11823CCP002				
Nome do Discente:	Mateus Curcino de Lima				
Título do Trabalho:	Exploração de estratégias para a classificação de fluxos de dados de imagens				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Ciência de Dados				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Fabíola Souza Fernandes Pereira - FACOM/UFU, Bruno Augusto Nassif Travençolo - FACOM/UFU, Elaine Ribeiro de Faria Paiva - FACOM UFU (Coorientadora), Ricardo Cerri - DC/UFSCAR, Elaine Parros Machado de Sousa - ICMC/USP e Maria Camila Nardini Barioni - FACOM UFU, orientadora do candidato.

Os examinadores participaram desde as seguintes localidades: Ricardo Cerri e Elaine Parros Machado de Sousa - São Carlos/SP, Fabíola Souza Fernandes Pereira - Monte Carmelo/MG. Os outros membros da banca e o aluno participaram da cidade Uberlândia/MG.

Iniciando os trabalhos a presidente da mesa, Prof.ª Dr.ª Maria Camila Nardini Barioni, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Doutor.

O competente diploma será expedido após cumprimento dos demais requisitos,

Agradecimentos

Agradeço primeiramente a Deus pela vida e por possibilitar a realização deste trabalho.

Às minhas orientadoras Prof.ª Dr.ª Maria Camila e Prof.ª Dr.ª Elaine Ribeiro pelos ensinamentos, paciência, incentivo e disponibilidade.

Aos colegas Alex e Yan pelas discussões e pelo auxílio na realização de experimentos que fizeram parte desta tese.

A todos os professores e funcionários do PPGCO, em especial ao Erisvaldo, pelo grande auxílio nas questões administrativas.

À minha esposa Nathália por todo o companheirismo em questões pessoais, de trabalho, acadêmicas, pelas contribuições com a tese e, principalmente, por todo o suporte em nosso lar.

À chegada da minha filha Lívia, que trouxe luz, alegria e esperança em momentos difíceis vividos durante toda a trajetória do doutorado, principalmente com a partida de pessoas muito especiais: Jovino, Sebastiana, Ana e Vanderlei *(in memoriam)*.

Aos meus pais, José Bastos e Alair, por todo o apoio durante o doutorado e por não medirem esforços para que eu chegasse até esta etapa da vida.

A todos os meus familiares e amigos, que auxiliaram diretamente ou indiretamente, mas de uma maneira inesquecível.

À CAPES pelo apoio financeiro durante a realização do trabalho.

A todos, o meu muito obrigado!

Resumo

A classificação de fluxos de dados de imagens apresenta vários desafios, por exemplo, a evolução de conceitos das classes conhecidas (*concept-drift*) e o surgimento de novas classes (*concept-evolution*). Embora muitos estudos tratem sobre a classificação de fluxos de dados de imagens, algumas características desse contexto não foram exploradas em conjunto nesses trabalhos, por exemplo: métodos de avaliação específicos para cenários de fluxos de dados, evolução do descritor de características das imagens, atualização do modelo de decisão considerando características de ambientes de aplicações reais e algoritmos de classificação capazes de lidar com dados de alta dimensão. O objetivo principal do trabalho descrito nesta tese é contribuir para a classificação de fluxos de dados de imagens nas etapas de classificação, atualização do modelo e avaliação considerando aspectos inerentes de cenários de aplicações reais. Para tanto, foi desenvolvido o *framework* EVISClass para a avaliação de algoritmos de classificação de fluxos de dados de imagens. Esse *framework* é capaz de considerar: ocorrência de *concept-drift* e *concept-evolution*, atrasos para rotular imagens (latência) e técnicas de aprendizado ativo para a seleção de instâncias a serem rotuladas. Com a utilização desse *framework* constatou-se que a latência possui forte influência na eficácia dos resultados. Além disso, observou-se que técnicas de aprendizado ativo podem contribuir para a seleção de um menor número de instâncias rotuladas, sem impactar de maneira significativa a eficácia do classificador. Por fim, foi desenvolvido o algoritmo HubISC para a classificação de fluxos de dados de imagens. Esse algoritmo incorpora o aspecto *hubness*, que é inerente de dados de alta dimensão. O algoritmo HubISC fornece também uma estrutura de sumarização de instâncias por meio da utilização dos *hubs*, que são instâncias de dados representativas. Além disso, essas instâncias são utilizadas no algoritmo como uma estratégia de aprendizado ativo. Os resultados dos experimentos com o algoritmo HubISC mostram o potencial do algoritmo em termos de desempenho preditivo e da quantidade de instâncias rotuladas em relação aos algoritmos comumente usados para a classificação de fluxos de dados de imagens.

Palavras-chave: Classificação de Fluxos de Dados de Imagens, Avaliação de Fluxos de Dados de Imagens, Aprendizado Ativo, *Hubness*

Abstract

The image data stream classification presents several challenges, for example, the evolution of concepts of known classes (*concept-drift*) and the emergence of new classes (*concept-evolution*). Although many studies deal with the image data stream classification, these studies did not explore some characteristics of this context together. For example, specific evaluation methods for data stream scenarios, the evolution of the image descriptor (*feature-evolution*), updating the decision model considering characteristics of real application environments, and classification algorithms capable of dealing with high dimensional data. The work described herein aims to contribute to the image data stream classification exploring the stages of classification, model update, and evaluation, considering inherent aspects of real application scenarios. Therefore, the EVISClass framework was developed for the evaluation of algorithms for image data stream classification. This framework can consider: the occurrence of *concept-drift* and *concept-evolution*, delays for labeling images (latency), and active learning strategies for selecting instances to be labeled. The use of this framework allowed us to observe that latency has a strong influence on the efficacy of the results. Furthermore, we observed that active learning strategies could contribute to the selection of a smaller number of labeled instances without significantly impacting the classifier's effectiveness. Finally, the HubISC algorithm for the image data stream classification was developed. This algorithm incorporates the *hubness* aspect, which is inherent in high-dimensional data. The HubISC algorithm also provides a structure for summarizing instances using *hubs*, which are representative data instances. Furthermore, these instances are used in the algorithm as an active learning strategy. The experiment results with the HubISC algorithm show the potential in terms of predictive performance and the number of labeled instances compared to commonly used algorithms for image data stream classification.

Keywords: *Image Data Stream Classification*, *Image Data Stream Evaluation*, *Active Learning*, *Hubness*

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

BoVW – *Bag-of-Visual Words*
CNN – *Convolutional Neural Network*
CPE – *CNN-based Prototype Ensemble*
EVISClass – *EValuation of Image data Stream Classifiers*
HOG – *Histogram of Oriented Gradients*
HubISC – *Hubness for Image data Stream Classification*
iCaRL – *Incremental Classifier and Representation Learning*
KNN – *K-Nearest Neighbors*
LBP – *Local Binary Patterns*
NCM – *Nearest Class Mean*
SIFT – *Scale Invariant Feature Transform*
SVM – *Support Vector Machine*
RNN – *Recurrent Neural Networks*

Lista de Algoritmos

Lista de Notações e Símbolos

A – Acurácia
B – Bloco de imagens
C – Conjunto de classes
d – Dimensionalidade do vetor de características $\vec{\boldsymbol{x}}$
D – Conjunto de imagens
$f()$ – Classificador
$\vec{\boldsymbol{h}}$ – *hub*
h_K – Limiar para definição dos *hubs*
$h_K(i)$ – Pontuação *hubness* de i
H – Conjunto de *hubs*
i – Imagem
K – Vizinhança
l – Latência
L – Limiar de distância entre duas instâncias de dados
m – Distância entre duas instâncias de dados
M – *F-measure*
n – Quantidade de imagens
n_C – Quantidade de classes
P – Probabilidade de i pertencer a uma determinada classe
S – Fluxo de dados de imagens
T – Tamanho do *buffer* Z
U – Incerteza de $f()$ em relação a classificação de uma determinada imagem
$\vec{\boldsymbol{x}}$ – Vetor de características de uma imagem i
X – Conjunto de imagens para treinamento inicial de $f()$
y – Rótulo de uma imagem i atribuído por $f()$
$\hat{y}$ – Rótulo real de uma imagem i
Z – *Buffer* de imagens
μ – Centróide

Sumário

Capítulo 1

Introdução

A manipulação de grandes quantidades de imagens tem se mostrado um aspecto importante para várias aplicações reais, por exemplo: segurança pública (AKCAY et al., 2018), diagnóstico médico (Karhan; Ergen, 2015), reconhecimento facial (ANUSHA et al., 2016) e sensoriamento remoto (BOUALLEG; FARAH, 2018). Nas últimas duas décadas, diferentes áreas da Computação, como Bancos de Dados e Mineração de Dados, têm trabalhado no desenvolvimento de estratégias visando a organização, a recuperação, a análise e a extração de informações úteis de conjuntos de imagens. Apesar dos resultados alcançados até o momento, ainda existem questões em aberto que precisam ser tratadas para que tarefas de gerenciamento e análise nesse domínio complexo de dados possam ser realizadas de maneira mais eficiente.

Dentre as tarefas de aprendizado de máquina relevantes para tarefas de gerenciamento e mineração de dados em conjuntos de imagens, destaca-se a classificação de imagens. A tarefa de classificação de imagens tem como objetivo atribuir rótulos a imagens a partir de um modelo de decisão treinado a partir de um conjunto de imagens rotuladas (THOMAS; PILLAI, 2019). A tarefa de classificação automática de imagens tem se mostrado desafiadora porque precisa lidar com diferentes questões, tais como: escolher a forma mais adequada para a representação de uma imagem, decidir como treinar o classificador, e definir como avaliar o desempenho do classificador considerando o cenário da aplicação alvo (SRIVASTAVA; RAJITHA; Agarwal, 2017). Esses desafios têm motivado a criação de algoritmos eficientes e eficazes para a classificação de imagens (WILLIAMS; LI, 2016).

Existem vários algoritmos propostos na literatura científica da área para a classificação de imagens, dentre eles, alguns desses trabalhos são apresentados em (LUO et al., 2018; BO; LU; WANG, 2018; BOUROUIS et al., 2019; ZHUANG et al., 2020; TEZCAN; KIRAS; BILGIN, 2022). Cada um desses trabalhos possui diferentes características, como o classificador empregado, o contexto considerado e a forma de representação das imagens utilizada. Um ponto em comum nesses trabalhos é que eles consideram um ambiente estático, isto é, um ambiente no qual a distribuição de probabilidade que gera os dados é

estacionária. Nesse caso, uma vez aprendido um modelo de decisão, esse não é alterado. Assim, esses trabalhos consideram que todas as classes do problema são previamente conhecidas e que um conjunto representativo de imagens rotuladas está disponível para treinamento do modelo.

Atualmente, com a alta disponibilidade de dispositivos de aquisição de imagens em diferentes aplicações, surgiu a necessidade do desenvolvimento de estratégias para lidar com fluxos de dados de imagens. Fluxos de dados consistem em massivas quantidades de dados, geradas em um curto espaço de tempo, de maneira contínua e potencialmente infinita (SILVA et al., 2013). Dentre os exemplos de aplicações de fluxos de dados de imagens estão presentes a classificação de objetos em tempo real (GANGINENI et al., 2019), reconhecimento de faces (AWAIS et al., 2019) e detecção de movimentos (GONG; SHU, 2020). Nesses exemplos de aplicações, deve-se utilizar estratégias para o treinamento online dos classificadores. Esse é um requisito importante uma vez que os classificadores agem em fluxos contínuos de dados, nos quais não se conhece todas as classes de imagens (*concept-evolution*) (GURJAR; CHHABRIA, 2015), não se tem o rótulo de todas as imagens para a atualização do modelo e os conceitos podem mudar ao longo do tempo (*concept-drift*) (JANARDAN; MEHTA, 2017).

Trabalhos recentes da literatura de classificação de imagens (WU et al., 2019; HU et al., 2021; JAISWAL, 2021; NAKATA et al., 2022; ZHENG; WEN, 2022) têm considerado a natureza dinâmica do problema no qual o modelo de decisão precisa ser constantemente atualizado. Entretanto, esses trabalhos avaliam tais modelos em cenários bem controlados supondo que todas as classes do problema são previamente conhecidas e que os rótulos estão sempre disponíveis para atualizar o modelo. Esse método de avaliação é inadequado para cenários de aplicações reais, nos quais diferentes classes de imagens chegam continuamente e não se dispõe de um especialista para rotular todas as imagens.

Tradicionalmente, os métodos utilizados para a classificação de imagens precisam de grandes quantidades de imagens rotuladas para o treinamento do classificador. O procedimento de rotular imagens pode ser caro e até mesmo impraticável em ambientes de aplicações reais (WANG et al., 2017). Para superar essa limitação, é possível considerar o emprego de técnicas de aprendizado ativo para diminuir o número de imagens rotuladas. Usando técnicas de aprendizado ativo, o algoritmo pode solicitar ao usuário o rótulo de imagens específicas (SETTLES, 2010), com o objetivo de rotular imagens mais significativas e obter resultados mais acurados com um menor número de imagens rotuladas. O principal desafio para as técnicas de aprendizado ativo, em ambientes de fluxos de dados de imagens, é selecionar de maneira eficiente as imagens mais significativas para serem rotuladas.

O uso de estratégias de aprendizado ativo também pode levar em consideração que em ambientes de fluxos de dados é comum que os rótulos não sejam fornecidos imediatamente. O atraso para o fornecimento de rótulos é conhecido na literatura como latência. O uso de latência permite aos métodos a simulação de ambientes de aplicações

reais, nos quais se tem a disposição, por exemplo, um usuário especialista para rotular as imagens. Alguns desafios para o emprego de latência na classificação de fluxo de dados de imagens são relacionados à taxa de atraso considerada e à disponibilidade do usuário especialista, pois são fatores que podem impactar a qualidade da classificação. Exemplos de trabalhos correlatos que consideram o fornecimento de rótulos com atraso para a atualização do modelo são descritos em (PARREIRA; PRATI, 2019; SOUZA et al., 2015).

Outro ponto a ser considerado na classificação de imagens é a forma de representação considerada. Os métodos de representação das imagens mais utilizados são baseados em *Bag-of-Visual Words* (NAPOLETANO, 2017) e *Deep Learning* (RAZAVIAN et al., 2014). No contexto de classificação de fluxos de dados de imagens, uma questão importante está relacionada à utilização de técnicas que permitam uma representação dinâmica das imagens (do inglês, *feature-evolution* - FE) (MASUD et al., 2013), considerando a existência de *concept-evolution* e *concept-drift*. Outro desafio para a representação de imagens é a alta dimensionalidade dos dados, isto é, os vetores gerados para a representação das imagens podem conter um alto número de atributos, causando a maldição da dimensionalidade (SAMET, 2005), que pode ter um impacto negativo em algoritmos baseados em distâncias.

Tradicionalmente, a questão da alta dimensionalidade tem sido tratada na literatura científica da área com a utilização de estratégias que procuram atenuar os efeitos da maldição da dimensionalidade, realizando a redução de dimensionalidade por meio de extração e de seleção de atributos (WANG et al., 2021). Outra maneira explorada para minimizar esse desafio, pode ser a utilização de descritores considerados estado da arte para imagens, por exemplo, as redes neurais convolucionais (do inglês, *Convolutional Neural Networks* - CNNs) (ALZUBAIDI et al., 2021; WANG et al., 2019). Esse descritor de características permite selecionar diferentes camadas de rede para gerar os vetores de características. As camadas mais internas da CNN geram vetores que representam mais detalhes das imagens, por exemplo: texturas, bordas e formas. Entretanto, esses vetores apresentam alta dimensionalidade. As camadas mais externas da CNN possibilitam a geração de vetores com menor dimensionalidade, porém os vetores apresentam níveis de detalhes inferiores, ou seja, algumas informações importantes das imagens podem ser perdidas.

Outro recurso considerado para o desenvolvimento de técnicas que permitem a classificação em bancos de dados de alta dimensão é a utilização de um aspecto inerente desse contexto, denominado *hubness* (FELDBAUER, 2019; WU et al., 2020b). O aspecto *hubness* consiste na tendência de algumas instâncias de dados, chamadas *hubs*, ocorrerem com maior frequência nas listas dos K-vizinhos mais próximos de outras instâncias (MANI et al., 2019). Até onde é de conhecimento do autor, apesar do potencial demonstrado, o uso de *hubs* foi explorado apenas no contexto de classificação de imagens para cenários estáticos. Os *hubs* estão localizados em regiões densas, com outras instâncias, o que pode

facilitar a propagação de rótulos para outras instâncias a partir do rótulo *hub*. Então, os *hubs* têm potencial para representar uma estrutura de sumarização das instâncias e indicar instâncias a serem rotuladas em processos de aprendizado ativo. Outro fator é que em ambientes de fluxos de dados é importante a utilização de técnicas de esquecimento de representantes (AGRAHARI; SINGH, 2021). Devido, principalmente, ao fenômeno *concept-drift*, é observada a necessidade de manter os representantes atualizados no modelo de decisão, removendo representantes que podem não refletir o atual momento. Dessa forma, os *hubs* também tem potencial para a identificação das instâncias representativas que sejam mais recentes. Por fim, outra informação que pode ser obtida por meio dos *hubs* é a identificação de ruídos, considerando instâncias de dados distantes das demais (TOMAšEV; BUZA, 2015).

1.1 Objetivos

O objetivo deste trabalho de doutorado é contribuir para a classificação de fluxos de dados de imagens nas etapas de classificação, atualização do modelo e avaliação considerando aspectos inerentes de cenários de aplicações reais. Neste trabalho, considera-se os seguintes aspectos para cenários de aplicações reais:

- Ocorrência dos fenômenos *concept-drift* e *concept-evolution*.
- Latência e disponibilidade limitada de um usuário especialista para fornecimento de rótulos;
- Alta dimensionalidade dos dados.

Para tanto, foram considerados os seguintes objetivos específicos:

- Avaliar a influência do uso de diferentes métodos de extração de características de imagens, estado da arte, na eficácia de classificadores de fluxos de dados de imagens, incluindo métodos capazes de fornecer representações dinâmicas.
- Desenvolver um método de avaliação para algoritmos de classificação de fluxos de dados de imagens, considerando aspectos de cenários de aplicações reais.
- Desenvolver um algoritmo para a classificação de fluxos de dados de imagens, baseado em *hubs*, que seja capaz de lidar com características desse contexto.

1.2 Hipótese

Considerando os pontos em aberto para a classificação de fluxos de dados de imagens e os objetivos de pesquisa descritos na Seção 1.1, a questão de pesquisa considerada nessa tese é: *como os aspectos inerentes de cenários de aplicações reais devem ser abordados no desenvolvimento de algoritmos de classificação de fluxos de dados de imagens, para*

que esses algoritmos sejam capazes de produzir resultados eficazes? Para tanto, foram exploradas as seguintes hipóteses de pesquisa:

- **H1**: o desempenho preditivo de algoritmos de classificação de fluxos de dados de imagens sofre a influência da quantidade de imagens e de classes de imagens utilizada na construção dos descritores de características.
- **H2**: os algoritmos de fluxos de dados de imagens da literatura científica da área, quando submetidos a cenários de aplicação reais, têm o seu desempenho preditivo degradado.
- **H3**: o uso de *hubs* como estrutura de sumarização de dados e como instâncias a serem rotuladas na etapa de aprendizado ativo leva a uma diminuição do número de instâncias rotuladas e em um melhor desempenho preditivo na classificação de fluxos de dados de imagens.

1.3 Contribuições

As principais contribuições dessa tese são:

- Avaliação da influência da quantidade de imagens e de classes de imagens utilizadas para a construção de descritores de características no desempenho preditivo de algoritmos de fluxos de dados de imagens.
- Desenvolvimento e disponibilização para a comunidade de um *framework* de avaliação de classificadores de fluxos de dados de imagens, que considera aspectos presentes em cenários de aplicações reais.
- Desenvolvimento e disponibilização para a comunidade de um classificador eficaz para fluxos de dados de imagens, que considera aspectos presentes em cenários de aplicações reais.

1.4 Organização da Tese

Esta tese está organizada como se segue. No Capítulo 2 são definidos os fundamentos teóricos. No Capítulo 3 são apresentados os trabalhos relacionados. O Capítulo 4 apresenta de maneira geral a proposta de classificação fluxos de dados de imagens desenvolvida neste trabalho. O Capítulo 5 apresenta um um estudo experimental com descritores de características para a representação de imagens na classificação de fluxos de dados imagens, que explora a hipótese de pesquisa **H1**. No Capítulo 6 é apresentado o *framework* de avaliação desenvolvido para a classificação de fluxos de dados de imagens, além de descrever experimentos realizados com recursos de *feature-evolution*. Nesse capítulo investigou-se a hipótese de pesquisa **H2**. No Capítulo 7 é apresentado o algoritmo desenvolvido, baseado em *hubs*, para a classificação de fluxos de dados de imagens, explorando a hipótese de

pesquisa **H3**. Por fim, no Capítulo 8 são apresentadas as conclusões do trabalho e os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os fundamentos teóricos relacionados ao trabalho descrito nesta tese. Formas de representação de imagens são descritas na Seção 2.1. A Seção 2.3 descreve a tarefa de classificação de imagens, com destaque para a classificação de fluxos de dados descrita na Seção 2.3.1. A Seção 2.3.2 descreve sobre métodos de aprendizado ativo. Os diferentes métodos de avaliação são discutidos na Seção 2.4, sendo que a Seção 2.4.1 descreve os métodos para a avaliação da classificação *batch* e a Seção 2.4.2 os métodos de avaliação para classificação de fluxos de dados 2.4.2. Por fim, a Seção 2.5 apresenta as considerações finais.

2.1 Representação de imagens

O processo para se classificar uma imagem envolve algumas etapas. Resumidamente, é necessário coletar a imagem, pré-processar, extrair as características e, finalmente, classificar (RZANNY et al., 2017). A etapa de extração de características é uma das principais, pois transforma a maneira de representação das imagens. Essas etapas básicas para a classificação de uma imagem estão ilustradas na Figura 1.

A partir da Figura 1 nota-se que a primeira etapa do processo de classificação de imagens consiste na coleta da imagem. Geralmente, essa coleta pode ser realizada por meio de aparelhos específicos, como é o caso de câmeras, satélites e equipamentos médicos. Geralmente, as imagens são representadas em arquivos digitais, que possuem matrizes de *pixels*. Após a aquisição da imagem, podem ser realizados procedimentos de pré-processamento, por exemplo, o redimensionamento da imagem para um tamanho específico, a alteração de cores e a remoção de partes do conteúdo (CHAKI; DEY, 2018). Antes de se realizar a classificação, deve-se transformar a maneira de representação das imagens para sua utilização nos algoritmos.

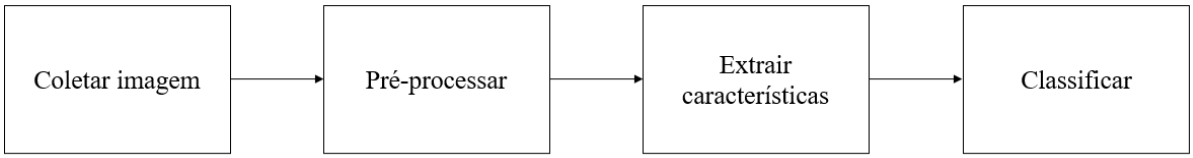

Figura 1 – Principais etapas para a classificação de uma imagem.

O processo de transformar a maneira de representação das imagens é chamado de extração de características. O resultado deste processo é a transformação da matriz de *pixels*, que representa uma imagem, em um vetor numérico $\vec{\boldsymbol{x}}$ responsável por descrever as características das imagens (ANDREU; MOLLINEDA; GARCíA-SEVILLA, 2009). O vetor pode ser representado por $\vec{\boldsymbol{x}} = x_1, ..., x_d$, onde x_i é cada uma das características das imagens obtidas pelos métodos de extração de características. Esses vetores de características são utilizados para comparar as imagens por meio da utilização de funções de distância ou medidas de similaridade (PESTOV, 2007). Algumas medidas de similaridade estão descritas na Seção 2.1.1.

Os métodos utilizados para a extração de características merecem atenção, pois por mais que sejam sofisticados, existem desafios para a representação das imagens. Um desafio é conhecido como *gap* semântico, que consiste na distância entre a representação da imagem e seu verdadeiro conceito semântico (LI et al., 2016). Outro desafio é a alta dimensionalidade dos dados, isto é, os vetores gerados para a representação das imagens podem conter um alto número de atributos, causando a maldição da dimensionalidade (SAMET, 2005). Um aspecto da maldição da dimensionalidade é a concentração de distâncias, que indica a tendência das distâncias serem quase iguais entre todas os vetores de características em alta dimensão. Essa característica pode ter um impacto negativo em qualquer algoritmo baseado em distâncias (PESTOV, 2007). Apesar desses desafios, entre os métodos utilizados para extração de características, existem duas grandes categorias de descritores de imagens: *Classical Feature Engineering*, também conhecida como descritores *Handcrafted*, e *Deep Learning Features* (NAPOLETANO, 2017).

2.1.1 Medidas de similaridade

Para calcular a similaridade entre duas imagens, pode-se utilizar uma função de distância para comparar os vetores de características $\vec{\boldsymbol{x}}$, obtidos pelos métodos de extração de características. Formalmente, a descrição das imagens é representada por um vetor de características (ou atributos) d-dimensional, onde d corresponde ao número de características de cada imagem. A escolha da função de distância a ser empregada irá depender do domínio de dados manipulado. As funções de distância da família *Minkowski* são comumente utilizadas para a comparação entre vetores de características de imagens (LI; DING; ZHANG, 2011). A Equação 1 apresenta a fórmula geral para as funções de distância de *Minkowski*:

$$m(\vec{\boldsymbol{x}}', \vec{\boldsymbol{x}}'') = \sqrt[p]{\sum_{j=1}^{d} \left(x_j' - x_j''\right)^p} \tag{1}$$

Na Equação 1, m representa a função de distância, $\vec{\boldsymbol{x}}'$ é o vetor de características da primeira imagem e $\vec{\boldsymbol{x}}''$ da segunda imagem comparada. Além disso, x_j' e x_j'' são os valores do j-ésimo atributo de $\vec{\boldsymbol{x}}'$ e $\vec{\boldsymbol{x}}''$, respectivamente. Geralmente, utiliza-se $p = 2$. A seguir, são apresentadas algumas funções de distância definidas a partir da fórmula geral de *Minkowski*:

- **Distância Euclidiana:** é considerada a distância mais utilizada para dados numéricos. Para dois vetores de características $\vec{\boldsymbol{x}}'$ e $\vec{\boldsymbol{x}}''$, no espaço d-dimensional, a distância Euclidiana entre eles é definida como:

$$m(\vec{\boldsymbol{x}}', \vec{\boldsymbol{x}}'') = \sqrt{\sum_{j=1}^{d} \left(x_j' - x_j''\right)^2} \tag{2}$$

- **Distância Manhattan:** também conhecida como distância *City-Block*. É definida como sendo a soma das distâncias de todos os atributos. Isto é, para dois vetores de características $\vec{\boldsymbol{x}}'$ e $\vec{\boldsymbol{x}}''$, no espaço d-dimensional, a distância entre eles é:

$$m(\vec{\boldsymbol{x}}', \vec{\boldsymbol{x}}'') = \sum_{j=1}^{d} \left|x_j' - x_j''\right| \tag{3}$$

  A distância de *Manhattan* é muito usada para lidar com atributos discretos, por exemplo, valores binários (FACELI; LORENA; GAMA, 2011).
- **Distância Máxima:** também é chamada de distância *Sup*. Ela é definida como sendo o valor máximo das distâncias dos atributos, isto é, para dois vetores de características $\vec{\boldsymbol{x}}'$ e $\vec{\boldsymbol{x}}''$, no espaço d-dimensional, a distância máxima entre eles é:

$$m(\vec{\boldsymbol{x}}', \vec{\boldsymbol{x}}'') = \max_{1 \leq j \leq d} \left|x_j' - x_j''\right| \tag{4}$$

Existem outras diversas medidas de similaridade propostas para outros diferentes domínios de dados. Uma visão geral sobre essas funções de distância pode ser encontrada em (COLLINS; OKADA, 2012) e (SAAD; KAMARUDIN, 2013).

2.1.2 *Classical Feature Engineering*

Na categoria *Classical Feature Engineering* existem diferentes critérios que podem ser utilizados para a descrição de uma imagem, por exemplo, cor, textura e forma (MIRMEHDI; XIE; SURI, 2009). Nesse contexto, alguns exemplos de descritores são: histograma de cor (NOVAK; SHAFER, 1992) e HOG (*Histogram of Oriented Gradients*) (JUNIOR et al., 2009). Além dessas técnicas, é possível descrever as imagens a partir de pontos de

interesse (*key points*). Um exemplo de descritor que considera esse conceito é o SIFT (*Scale Invariant Feature Transform*) (LOWE, 2004). Uma abordagem que pode utilizar este método é a BoVW (*Bag-of-Visual Words*) (NAPOLETANO, 2017).

A abordagem BoVW (NAPOLETANO, 2017) é inspirada no método *Bag-of-Words* utilizado para a categorização de textos. Nesse caso, um documento é representado pela frequência das palavras extraídas no texto. Já na abordagem BoVW, uma imagem é representada por palavras visuais. Essa abordagem apresenta algumas vantagens para a descrição de imagens, por exemplo, flexibilidade e robustez. Esses fatores podem ser percebidos, por exemplo, por meio da obtenção de resultados invariantes independente da rotação aplicada a essa imagem. Devido a esses fatores, a abordagem BoVW recebe importante atenção da literatura científica da área (CSURKA et al., 2004; HOU; KANG; QI, 2010; KUMAR et al., 2017; SILVA et al., 2018; LIU; CAI; ZHANG, 2018; SRIVASTAVA; RAJITHA; Agarwal, 2017; SAINI; SUSAN, 2019; SRIVASTAVA; BAKTHULA; AGARWAL, 2019; VELOSO, 2017) para a descrição de imagens.

De maneira geral, os descritores de características de imagens baseados em BoVW consistem em identificar pontos-chave em uma imagem e atribuí-los a um conjunto de palavras visuais. Então, um histograma de frequência de palavras visuais é utilizado para representar uma imagem. Duas imagens são ditas semelhantes se os histogramas forem semelhantes de acordo uma medida de similaridade (SILVA et al., 2018). A Figura 2 apresenta as principais etapas do método BoVW.

Conforme pode ser verificado na Figura 2, no método BoVW cada imagem é representada por um vetor $\vec{\boldsymbol{x}}$, que é composto por um histograma de d palavras visuais. O vocabulário de palavras visuais é construído com base nos pontos-chave das imagens disponíveis. Esse procedimento está ilustrado nas etapas (i) e (ii) da Figura 2(A). Após a identificação dos pontos-chave das imagens, por exemplo, a partir do algoritmo SURF, deve-se criar o vocabulário de palavras visuais. Diferentes abordagens podem ser aplicadas nessa etapa. A abordagem aleatória é a mais simples, pois as palavras visuais são escolhidas aleatoriamente a partir dos pontos-chave. Essa abordagem apresenta rapidez, mas geralmente não produz bons resultados, já que nenhuma informação adicional foi considerada para a definição das palavras visuais (VELOSO, 2017).

Estratégias de aprendizado não supervisionado, como é caso de algoritmo de agrupamento de dados podem ser aplicadas para a criação do vocabulário de palavras visuais. Para tanto, o algoritmo *k-means* é uma abordagem comumente considerada (CHATFIELD et al., 2011). O processo de agrupamento pode ser observado nas etapas (iii) e (iv) da Figura 2(A). Nessa ilustração, considera-se 3 grupos principais, que recursivamente podem formar subgrupos. Os centróides dos grupos representam as palavras visuais a serem consideradas, como pode ser visualizado no histograma da Figura 2(B).

Um ponto importante sobre o processo de agrupamento é a definição da quantidade de grupos e, consequentemente, de palavras visuais que serão consideradas para a criação

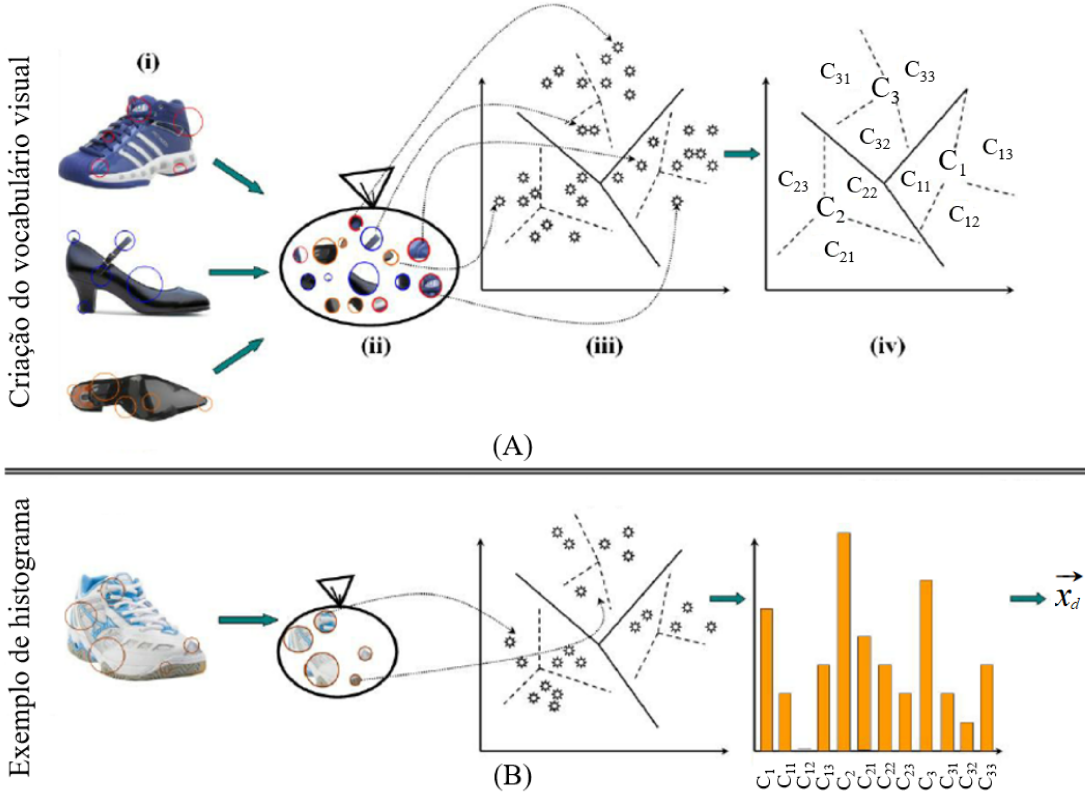

Figura 2 – Exemplo do uso de BoVW para descrição de imagens. (A) Criação do vocabulário visual. (B) Criação de um histograma de palavras visuais. Adaptado de (TOMASIK; THIHA; TURNBULL, 2009)

do vocabulário visual. Sugere-se a utilização de uma quantidade considerada suficiente para que seja possível realizar a distinção entre os conteúdos das imagens. Entretanto, essa quantidade também não deve ser exagerada, pois pode dificultar a distinção de partes irrelevantes das imagens, como é o caso da presença de ruídos (CSURKA et al., 2004).

Com a criação do vocabulário de palavras visuais, uma imagem i é representada por um vetor $\vec{\boldsymbol{x}} = x_1, ..., x_d$, onde x_i pode corresponder à frequência de cada uma das palavras visuais. Para a comparação entre duas imagens, que foram descritas com o método BoVW, pode-se utilizar funções de distância ou medidas de similaridade que analisam os vetores $\vec{\boldsymbol{x}}$.

2.1.3 *Deep Learning Features*

Considerar apenas descritores *Handcrafted* pode não ser suficiente para a descrição de imagens, por exemplo, devido a necessidade de lidar com o desafio do *gap* semântico (LI et al., 2016) para a representação das imagens. De forma a investigar alternativas para esse problema, a categoria de descritores *Deep Learning Features* pode ser utilizada (NAPOLETANO, 2017).

A categoria de descritores *Deep Learning Features* considera o conceito de *Deep Learning*, que consiste em um ramo do aprendizado de máquina que utiliza um conjunto de

algoritmos para modelar problemas, a partir de várias camadas de processamento. Entre as tarefas compatíveis com a aplicação de *Deep Learning* está a extração de características de imagens (GOODFELLOW; BENGIO; COURVILLE, 2016). As estratégias baseadas em *Deep Learning* mais comumente utilizadas consideram CNNs *(Convolutional Neural Networks)* e RNNs *(Recurrent Neural Networks)*. Geralmente, as CNNs são utilizadas para a extração de características de imagens em cenários de fluxos de dados. Já as RNNs são mais adequadas para a utilização em cenários estáticos (MIEBS et al., 2020).

Modelos baseados em *Deep Learning* são compostos por múltiplas camadas de processamento. Uma arquitetura típica de uma CNN, para a representação de imagens, consiste em uma ou mais camadas convolucionais, seguidas por camadas de *pooling*. Geralmente, as camadas *pooling* simplificam as informações das saídas das camadas convolucionais, isto é, podem diminuir a dimensionalidade dos vetores de características. As últimas camadas da CNN são chamadas de totalmente conectadas. O resultado da última camada totalmente conectada é a saída da CNN (GOO et al., 2016). Resumidamente, de maneira geral, as CNN podem ser entendidas como uma rede neural que possui diversas camadas ocultas, com o objetivo de representar problemas complexos em etapas mais simples.

Para a extração de características de imagens, um conjunto de imagens X é utilizado como entrada da rede. As imagens são fornecidas em sua forma bruta (matriz de *pixels*). Na sequência, cada imagem $i \in X$ passa pelas camadas convolucionais. A cada camada convolucional, a rede aprende mais características das imagens. As primeiras camadas convolucionais são capazes de aprender características como bordas e texturas. As camadas posteriores aprendem recursos padrões mais complexos, como partes de objetos. Por fim, as camadas totalmente conectadas podem conectar as características aprendidas nas camadas anteriores e fornecer os vetores $\vec{\boldsymbol{x}}$ com as características das imagens. É importante destacar que a quantidade de camadas de uma CNN é particular de cada arquitetura. Alguns exemplos tradicionais de arquiteturas de CNN são: LeNet, AlexNet, GoogLeNet, DenseNet e VGG16. Além do número de camadas, cada arquitetura pode possuir suas peculiaridades, como o número de parâmetros considerados e as operações realizadas pela rede, o que pode interferir no seu desempenho em termos de eficácia e eficiência (GU et al., 2018).

Para a comparação entre duas imagens pode-se utilizar os vetores de características $\vec{\boldsymbol{x}}$ formados pela CNN e compará-los por meio da utilização de funções de distância ou medidas de similaridade (veja Seção 2.1.1). Para esse procedimento, pode-se utilizar os vetores de características de diferentes camadas da rede ou combinar os vetores dessas camadas. Geralmente, utiliza-se a camada conhecida como *flatten* para esse propósito. As camadas mais externas da rede, geralmente, têm menor dimensionalidade, o que pode facilitar a comparação entre dois vetores de características. Entretanto, essas camadas representam características mais simples das imagens. Já as camadas mais internas da rede

representam características mais robustas das imagens, o que pode ser importante para a distinção das imagens, mas também pode significar que os vetores de características tenham maior dimensionalidade (SCHUCH, 2019). Esse fato deve ser levado em consideração, já que pode ter influência em relação a maldição da dimensionalidade para a comparação entre duas imagens a partir de funções de distância ou medidas de similaridade (SAMET, 2005) (veja Seção 2.1.1). Para lidar com a maldição da dimensionalidade uma das alternativas é considerar o aspecto *hubness*, detalhado na Seção 2.2.

2.2 Aspecto *Hubness*

Hubness é um aspecto da *maldição da dimensionalidade*. *Hubs* são instâncias de dados que aparecem na lista de vizinhos mais próximos de um grande número de outras instâncias (FELDBAUER, 2019). Para analisar um conjunto de dados com o aspecto *hubness* é necessário calcular a pontuação *hubness* ($h_K(i)$) de cada instância. Além disso, podem ser considerados os conceitos de *hubs* e *anti-hubs*.

- **pontuação *hubness***: seja $X = \{i^1; ...; i^n\}$ um conjunto de instâncias de dados, $h_K(i)$ representa o número de K-ocorrências de instâncias $i \in X$, isto é, o número de vezes que i ocorre na listagem dos K-vizinhos mais próximos de outras instâncias de dados pertencentes a X. Valores comumente utilizados para K estão no intervalo [5, 20] (TOMASEV et al., 2014);
- ***hubs***: correspondem às instâncias de dados $i \in X$ que aparecem, notavelmente, em muitas listagens de K-vizinhos mais próximos das demais instâncias de dados, isto é, possuem $h_K(i)$ significativamente acima da média;
- ***anti-hubs***: correspondem às instâncias de dados $i \in X$ que não aparecem em praticamente nenhuma listagem de K-vizinhos mais próximos das demais instâncias de dados, ou seja, possuem $h_K(i)$ extremamente baixo, ou até mesmo, $h_K(i) = 0$;
- ***good-hubness***: é um conceito usual em trabalhos de classificação, quando os rótulos das instâncias de dados estão disponíveis. Esse conceito representa o número de K-ocorrências de uma instância $i \in X$ na lista de vizinhos mais próximos de outras instâncias de dados com o mesmo rótulo de i (TOMASEV et al., 2011);
- ***bad-hubness***: de maneira oposta ao conceito *good-hubness*, o conceito *bad-hubness* representa o número de K-ocorrências de uma instância $i \in X$ na lista de vizinhos mais próximos de outras instâncias de dados com o rótulo diferente de i. A pontuação *hubness* pode ser calculada por meio da soma das pontuações *good-hubness* e *bad-hubness*.

Tem sido demonstrado que o conceito *hubness* consiste em uma propriedade inerente de grande parte dos conjuntos de dados de alta dimensão, não sendo peculiar de conjuntos de dados específicos. O aspecto *hubness* pode indicar uma boa referência de pontos de centralidade (MANI et al., 2019). Os principais *hubs* podem ser usados efetivamente como

representantes para guiar o processo de atribuição de rótulos de imagens, de maneira similar a como ocorre nas técnicas que consideram outros tipos de representantes, por exemplo, centróides. A Figura 3 ilustra essa questão, onde o losango é o centróide da classe representada em preto, o círculo vermelho corresponde ao medóide e os *hubs* são destacados em verde (considerando $K = 3$).

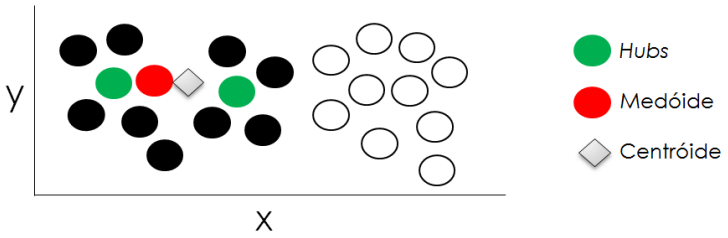

Figura 3 – Comparação *Hub vs.* Medóide *vs.* Centróide. Adaptado de (TOMASEV et al., 2014).

A utilização do conceito *hubness* tem sido observada em muitas tarefas, tais como: agrupamento de dados, recuperação da informação, classificação de imagens e detecção de ruídos (FELDBAUER, 2019). O trabalho descrito em (MANI et al., 2019) realizou um estudo do potencial do aspecto *hubness* em técnicas de classificação, agrupamento e recuperação de informações. Os trabalhos (FELDBAUER, 2019; WU et al., 2020b) consideraram o aspecto *hubness* para a classificação de conjuntos de dados de outros contextos. Além disso, em (TOMAŠEV et al., 2014) o aspecto *hubness* foi utilizado para a classificação de imagens em cenários estáticos. Até onde é de conhecimento do autor, o aspecto *hubness* não foi aplicado na classificação de fluxos de dados de imagens.

2.3 Classificação de imagens

A classificação é uma das técnicas de mineração de dados mais utilizadas (BIFET; READ, 2018), sendo considerada em diversas aplicações, tais como: redes de sensores, análise de mercado e perfis em mídias sociais (STEFANOWSKI; BRZEZINSKI, 2017). Entre essas categorias de aplicações, destaca-se a classificação de imagens, por exemplo, em tarefas para reconhecimento facial (YU et al., 2019), sensoriamento remoto (LI et al., 2019) e diagnóstico médico (CAO et al., 2019). Além disso, ressalta-se que a produção de imagens digitais pode ser realizada por diversas fontes, por exemplo: câmeras, smartphones, dispositivos médicos, drones e satélites.

Em termos gerais, dada uma lista de classes, a classificação de imagens procura prever a qual classe uma nova imagem pode pertencer. Formalmente, o problema de classificação pode ser formulado da seguinte forma: dado um conjunto de imagens da forma $(\vec{\boldsymbol{x}}, y)$, onde $\vec{\boldsymbol{x}} = x_1, ..., x_d$ é um vetor de valores de características que descrevem uma imagem e y é uma classe discreta de um conjunto C com n_C diferentes classes. O classificador cria um modelo $y = f(\vec{\boldsymbol{x}})$ para prever as classes y de futuros exemplos de

imagens. Por exemplo, $\vec{\boldsymbol{x}}$ poderia ser a descrição de uma imagem médica e y a classe de diagnóstico.

Por muito tempo, a classificação de imagens focou em cenários estáticos com pequenos conjuntos de dados. Nesse contexto, todas as imagens estão disponíveis e todas as classes de imagens são conhecidas. Esse cenário de classificação de dados é conhecido na literatura como classificação *batch* (STEFANOWSKI; BRZEZINSKI, 2017). Nesse cenário, como todas as classes de imagens do conjunto C são previamente conhecidas, uma vez criado o modelo de decisão $f()$, não existe a necessidade de reconstruí-lo, pois novas classes de imagens não surgirão. Exemplos tradicionais de algoritmos de classificação para o cenário *batch* são: *KNN (K-Nearest Neighbors)*, SVM *(Support Vector Machine)*, *Naive Bayes*, modelos baseados em árvores de decisão e redes neurais (SHI et al., 2019).

A maioria dos trabalhos da literatura para classificação de imagens é dedicada à classificação *batch*. No entanto, no contexto atual, com a evolução da tecnologia, principalmente dos dispositivos de armazenamento e de captura de imagens, é comum que várias imagens sejam coletadas em curto espaço de tempo e de forma contínua, constituindo um ambiente dinâmico, com fluxos de dados de imagens. Além disso, existem características particulares desse contexto, como o surgimento de novas classes de imagens no decorrer do fluxo (SILVA et al., 2013), o que pode provocar a necessidade de utilizar algoritmos adequados para esse contexto, com procedimentos diferentes em relação a classificação *batch*, como atualizar $f()$ de maneira online. A Seção 2.3.1 descreve a classificação de fluxos de dados.

2.3.1 Classificação de fluxos de dados

Atualmente, com a alta disponibilidade de dispositivos de aquisição de imagens em diferentes aplicações, surgiu a necessidade do desenvolvimento de estratégias para lidar com fluxos de dados de imagens. Fluxos de dados consistem em massivas quantidades de dados, geradas em curto espaço de tempo, de maneira contínua e potencialmente infinita (SILVA et al., 2013). Formalmente, um fluxo de dados de imagens S, é uma sequência massiva de imagens i^1, i^2, ..., i^n, potencialmente infinita, que chega nos *timestamps* t^1, t^2, ... ,t^n (FARIA et al., 2016), sendo cada imagem i^i descrita por um vetor de características $\vec{\boldsymbol{x}}^i$.

Dentre os exemplos de aplicações de fluxos de dados de imagens estão presentes a classificação de objetos em tempo real (GANGINENI et al., 2019), reconhecimento de faces (AWAIS et al., 2019) e detecção de movimentos (GONG; SHU, 2020). Nessas aplicações é comum que a massa de dados produzida seja grande demais para caber na memória principal e até mesmo em dispositivos de armazenamento secundário. Os algoritmos de fluxos de dados de imagens devem levar em consideração as seguintes restrições (SILVA et al., 2013):

1. As imagens chegam continuamente;
2. Não há controle sobre a ordem que as imagens devem ser processadas;
3. O tamanho de um fluxo é potencialmente ilimitado;
4. As imagens devem ser descartadas depois de terem sido processadas. Na prática, pode-se armazenar parte dos dados por um determinado período, usando um mecanismo de esquecimento para descartá-la posteriormente;
5. A frequência de geração de imagens é desconhecida, isto é, pode mudar ao longo do tempo.

A Tabela 1 apresenta uma comparação entre as características da classificação *batch* e da classificação de fluxos de dados. Nota-se que na classificação *batch* é possível passar várias vezes pelos dados, ou seja, realizar a leitura dos dados quantas vezes forem necessárias. Já na classificação de fluxos de dados isso não é possível, geralmente, realize-se uma única leitura dos dados. Na manipulação de fluxos de dados é comum lidar com limitações de *hardware*, por exemplo, a quantidade de memória disponível para uso, o que pode impactar na necessidade de descartar dados após o processamento ou de utilizar mecanismos de compactação. Outra diferença entre as formas de classificação é relação ao tempo de resposta dos algoritmos, na classificação de fluxos de dados muitas vezes a resposta deve ser em tempo real, por exemplo, em sistemas de reconhecimento facial. Além disso, o treinamento dos algoritmos de classificação em cenários *batch* é realizado uma única vez, já em ambientes dinâmicos pode ser necessário realizar o treinamento de maneira online (NGUYEN; WOON; NG, 2015).

Tabela 1 – Comparação entre as características da classificação *batch* e da classificação de fluxos de dados.

	Classificação *batch*	Classificação de fluxos de dados
Leitura dos dados	Múltiplas	Limitada
Tempo de resposta	Indefinido	Tempo real
Aprendizado	Único	Online

Da mesma forma que na classificação *batch*, o objetivo da classificação em fluxos de dados de imagens é predizer a classe das imagens não rotuladas. O problema de classificação de fluxos de dados de imagens pode ser definido formalmente como: a partir de um conjunto inicial X de vetores de características de imagens rotuladas, o classificador cria um modelo $f()$ para prever as classes $y \in \{c^1, c^2, ..., c^{nC}\}$ de novas imagens i que chegam em fluxo de imagens S. No entanto, em ambientes dinâmicos pode não ser adequado treinar o modelo $f()$ somente uma vez, pois existem fenômenos relacionados a fluxos de dados, por exemplo, *concept-drift* (JANARDAN; MEHTA, 2017) e *concept-evolution* (GURJAR; CHHABRIA, 2015).

2.3.1.1 *Concept-drift*

O fenômeno *concept-frift* ocorre quando as distribuições de dados mudam ao longo do tempo de forma inesperada e imprevisível (JANARDAN; MEHTA, 2017), o que significa que na classificação de fluxos de imagens o perfil das imagens, mesmo de classes já conhecidas, pode mudar em relação às imagens que foram utilizadas para o treinamento do algoritmo de classificação, o que pode refletir na qualidade dos resultados apresentados pelo classificador. Dessa forma, o modelo de decisão $f()$ não pode ser treinado uma única vez, já que os conceitos aprendidos no treinamento inicial podem não ser capazes de classificar corretamente as novas imagens disponíveis no fluxo de dados de imagens. Portanto, o modelo de decisão $f()$ deve ser treinado online com o fluxo de imagens S, para tratar os novos conceitos existentes.

O primeiro desafio relacionado ao fenômeno *concept-frift* é detectar a sua ocorrência. Após a detecção, deve-se adaptar o modelo de decisão em relação as novas imagens disponíveis. Contudo, essas tarefas não são triviais. Geralmente, duas estratégias são usadas para adaptar o modelo de decisão para tratar *concept-drift*: *Blind* e *Informed*. A primeira estratégia não verifica se realmente houve a ocorrência de *concept-drift*, o modelo de decisão é atualizado em intervalos regulares de tempo por meio das últimas imagens rotuladas disponíveis. Já a segunda estratégia atualiza o modelo de decisão no momento que se detecta a ocorrência de *concept-drift* (GAMA et al., 2014).

2.3.1.2 *Concept-evolution*

Ao contrário da classificação *batch*, na classificação de fluxos de dados é comum que não se conheça previamente todas as classes, já que nem todas as classes são conhecidas no treinamento inicial do algoritmo de classificação e novas classes podem aparecer ao longo do fluxo de dados (GURJAR; CHHABRIA, 2015). Por exemplo, o algoritmo de classificação pode ter sido treinado com n_C classes de imagens, mas ao longo do fluxo de imagens podem surgir w novas classes. Esse fenômeno é conhecido como *concept-evolution*. O termo *Open-set* também é comumente utilizado na literatura científica da área para se referir a conjuntos de dados para os quais não se conhece todas as classes.

Para lidar com o fenômeno *concept-evolution* os algoritmos de classificação de fluxo de dados de imagens devem ser capazes de detectar novas classes assim que elas aparecerem no fluxo de dados e atualizar o modelo de decisão. Nesse contexto, o modelo de decisão $f()$ não pode ser treinado uma única vez, já que no treinamento inicial n_C classes estão disponíveis e no teste do modelo de decisão podem aparecer $n_C + w$ classes, isto é, o modelo $f()$ deve atribuir um rótulo y' a um vetor de características $\vec{\boldsymbol{x}}$, sendo y' $\in \{c^1, c^2, ..., c^{n_C}, c^{(n_C)+1}, ..., c^{(n_C)+w}\}$. Portanto, o modelo de decisão $f()$ deve ser treinado de maneira online com o fluxo de imagens S, para considerar também as w novas classes. No entanto, para a atualização online do modelo $f()$ é necessário que os rótulos reais das

imagens das novas classes sejam disponibilizados, mas a entrega desses rótulos pode sofrer a influência da latência, descrita na Seção 2.3.1.3.

2.3.1.3 Latência

Os fenômenos *concept-drift* e *concept-evolution* podem afetar negativamente o desempenho dos algoritmos de classificação, pois caso esses fenômenos não sejam tratados o modelo de decisão pode não ser capaz de predizer corretamente o rótulo das imagens que surgem no fluxo de dados. Dessa forma, se torna necessário adaptar o modelo de decisão com as novas imagens, assim que os rótulos verdadeiros estiverem disponíveis.

Geralmente, os algoritmos de classificação de fluxos de dados assumem que os rótulos das imagens estarão disponíveis imediatamente, sem qualquer atraso. Contudo, essa suposição não é válida em muitos ambientes de aplicações reais. Nesses ambientes, existe um intervalo de tempo entre a requisição do rótulo da imagem e a sua disponibilidade. Esse atraso para a informação do rótulo é conhecido como latência (SOUZA et al., 2015). Os diferentes intervalos de latência são definidos da seguinte forma:

- (A) Latência Nula: nesse cenário os rótulos das imagens são disponibilizados imediatamente.
- (B) Latência Extrema: nesse cenário os rótulos das imagens nunca serão disponibilizados;
- (C) Latência Intermediária: nesse cenário os rótulos das imagens serão disponibilizados em algum momento futuro desconhecido. Além disso, nesse cenário os rótulos das imagens podem chegar em uma ordem diferente em relação a que foram apresentadas no fluxo de dados.

A latência nula é o melhor cenário para situações que é necessário lidar com os fenômenos *concept-drift* e *concept-evolution*, pois é possível realizar a atualização do modelo de decisão imediatamente. Contudo, sabe-se que essa suposição é inadequada para vários cenários de aplicações reais. A latência extrema seria o pior caso, pois com a indisponibilidade de rótulos é inviável atualizar o modelo de decisão de maneira online, o que pode impactar severamente a qualidade dos resultados apresentados pelo classificador. O caso mais compatível com cenários reais é a existência de latência intermediária, pois os rótulos serão disponibilizados em algum momento futuro. O tempo para a informação dos rótulos pode depender da disponibilidade de um usuário especialista. Para auxiliar o trabalho do usuário especialista, pode-se considerar métodos de aprendizado ativo, que estão descritos na Seção 2.3.2.

2.3.2 Aprendizado Ativo

Os algoritmos utilizados para a classificação de imagens, geralmente, precisam de uma grande quantidade de imagens rotuladas para o treinamento do classificador.

Entretanto, o atendimento dessa necessidade pode ser inviável em cenários de aplicações reais. Para diminuir o número de imagens rotuladas e também a forte dependência de usuários especialistas pode-se utilizar métodos de aprendizado ativo para a seleção de imagens mais informativas. Além de reduzir a carga de trabalho na obtenção dos rótulos, o aprendizado ativo identifica um conjunto de imagens capaz de treinar um classificador (WU et al., 2020a). Portanto, o aprendizado ativo pode ser uma solução muito útil para trabalhos de classificação de imagens que possuem um alto número de imagens, que é o caso da classificação de fluxos de dados de imagens.

Um dos principais desafios para a utilização de aprendizado ativo é identificar a estratégia de seleção de imagens mais adequada para o problema abordado, isto é, a estratégia que pode alcançar o melhor desempenho do classificador, mas utilizando o menor de conjunto de imagens possível, a partir da seleção das imagens mais representativas (ZHU et al., 2010). A maior parte dos métodos de aprendizado ativo são voltados para ambientes estáticos (classificação *batch*). Entre as estratégias de seleção mais populares, pode-se citar: amostragem por incerteza, *query-by-committee*, expectativa de mudança no modelo, redução do erro estimado e métodos de densidade e peso (SETTLES, 2010).

No cenário *batch*, a estratégia amostragem por incerteza solicita o rótulo das imagens mais incertas com relação à atribuição do seu rótulo. A estratégia *query-by-committee* considera a votação de um comitê de classificadores treinados, as imagens a serem rotuladas são as que apresentam maiores discordâncias entre os classificadores do comitê. A estratégia expectativa de mudança no modelo seleciona para rotular as imagens que trariam as maiores mudanças no modelo do classificador. A estratégia redução do erro estimado seleciona as imagens que minimizam o erro do classificador. Por fim, as estratégias de densidade e peso solicitam o rótulo das imagens de regiões mais densas do espaço.

Para ambientes dinâmicos, os métodos de aprendizado ativo mais populares são baseados em estratégias que consideram: aleatoriedade, incerteza na classificação, incerteza variável, incerteza com aleatoriedade, incerteza com custo e tempo (ŽLIOBAITė et al., 2014).

Estratégia Aleatória

A estratégia aleatória estabelece que a decisão de requisitar o rótulo de uma imagem é definida aleatoriamente. Para tanto, realiza-se o sorteio de uma probabilidade P, sendo $P = \{z \in \mathbb{R} : 0 \leq z \leq 1\}$. Então, verifica-se o valor de P em relação a uma constante b, utilizada como *budget*. Geralmente, o valor padrão de b é 0.5. No entanto, pode-se variar esse valor. Com valores maiores de b, mais rótulos podem ser solicitados, com valores menores, a tendência é menos rótulos sejam solicitados. Apesar de ser uma estratégia de simples implementação, não necessariamente as imagens mais representativas serão selecionadas para a informação do rótulo (ŽLIOBAITĖ et al., 2011).

Estratégia de Incerteza

A estratégia baseada em incerteza é umas das mais populares em técnicas de aprendizado ativo (SETTLES, 2010). Essa estratégia requisita os rótulos das imagens que o classificador apresenta menos confiança em relação a atribuição do rótulo (ŽLIOBAITĖ et al., 2011). Para tanto, pode-se calcular a probabilidade P das imagens pertencerem a cada classe. A partir do cálculo de P, determina-se o nível de incerteza U na classificação. Então, deve-se comparar U em relação a um limiar θ, que estabelece o grau de incerteza para a solicitação dos rótulos. Caso $U \geq \theta$ solicita-se o rótulo da imagem. Dessa forma, quanto menor o valor de θ, a tendência é que mais rótulos sejam solicitados. Além da incerteza, é possível utilizar outros critérios para a seleção das imagens, como ganho de informação e entropia

Estratégia de Incerteza Variável

A estratégia de incerteza variável é baseada na estratégia de incerteza tradicional. Entretanto, pode-se ajustar o valor do limiar θ no decorrer do fluxo de dados de imagens. O valor de θ pode se contrair ou expandir, em relação a um valor s, de acordo com a quantidade de rótulos solicitados (ŽLIOBAITĖ et al., 2011). Quando um rótulo é solicitado, o valor de θ aumenta, isto é, $\theta = \theta + s$. Já quando se passa muito tempo sem que nenhum rótulo seja solicitado, a tendência é que o valor de θ diminua, ou seja, $\theta = \theta - s$, para facilitar que o grau de incerteza seja compatível com o limiar estabelecido.

Estratégia de Incerteza com Aleatoriedade

A estratégia de incerteza com aleatoriedade também é inspirada na estratégia de incerteza tradicional. Assim como a estratégia de incerteza variável, o limiar pode sofrer ajustes no decorrer do fluxo de dados de imagens. Mas a diferença é que essa alteração não depende, necessariamente, da quantidade de rótulos solicitados. Além disso, a alteração do limiar θ será a partir da multiplicação do seu valor em relação a um valor aleatório δ. Essa alteração pode ocorrer a cada nova imagem que chega no fluxo de dados (ŽLIOBAITĖ et al., 2011). As alterações do limiar θ, provocada pelas estratégias de incerteza variável e incerteza com aleatoriedade, podem favorecer a detecção de *concept-drift* e *concept-evolution*, já que manter somente um valor de θ, no decorrer do fluxo de dados de imagens, pode ser insuficiente para a análise de incerteza na classificação de imagens que possuem mudanças de conceitos ou de imagens pertencentes a novas classes.

Estratégia de Incerteza com Custo

A estratégia de incerteza com custo considera o grau de incerteza na classificação, mas também leva em consideração o custo para obtenção do rótulo (PARREIRA; PRATI,

2019). Nesse caso, imagens com probabilidade P semelhantes de pertencerem a duas ou mais classes, podem ser mais complexas para a atribuição de um rótulo, o que pode comprometer a disponibilidade do usuário especialista para o fornecimento dos rótulos solicitados na sequência.

Estratégia de Tempo

Algumas estratégias de aprendizado ativo, utilizadas em contextos dinâmicos, podem considerar a latência para a solicitação dos rótulos. Por exemplo, pode-se selecionar as imagens de acordo com o tempo de chegada no fluxo de dados, ou seja, selecionar as instâncias mais antigas ou mais recentes para o fornecimento dos rótulos. Além disso, é possível realizar a combinação de diferentes estratégias, considerando o tempo, a incerteza e o custo, simultaneamente (PARREIRA; PRATI, 2019).

2.3.3 Algoritmos para classificação de fluxos de dados

Com a chegada constante de instâncias de dados, além dos desafios *concept-drift*, *concept-evolution* e latência pode ser inviável utilizar os mesmos algoritmos de classificação considerados para a classificação *batch* na classificação de fluxos de dados, pois existe a necessidade da atualização online do modelo de decisão. Além disso, na classificação de fluxos de dados de imagem podem ocorrer mais desafios, como: o *gap* semântico e a maldição da dimensionalidade, descritos na Seção 2.1. Algumas adaptações em algoritmos utilizados no cenário *batch* foram realizadas para adequá-los às restrições impostas pela classificação de fluxos de dados. Entre essas adaptações, destaca-se a utilização de janelas para o processamento das imagens e, consequentemente, para a atualização online do classificador. As estratégias de janelas mais utilizadas são: *Landmark Window*, *Sliding Window* e *Fading Window* (NGUYEN; WOON; NG, 2015).

Landmark Window

A estratégia *Landmark Window* considera todo o fluxo de dados, desde o instante inicial de tempo 1 até o instante de tempo atual (NGUYEN; WOON; NG, 2015). Nesse cenário, todas as imagens na janela são igualmente importantes, pois não existe diferença entre imagens passadas ou presentes no fluxo de imagens. No entanto, com a ocorrência de *concept-drift* e *concept-evolution* pode ser necessário enfatizar as imagens mais recentes. Para tanto, pode-se aplicar outras estratégias de janelas, como *Sliding Window* e *Fading Window*.

Sliding Window

Na estratégia *Sliding Window* o foco é somente nas q imagens mais recentes, as demais imagens são ignoradas. O desempenho do classificador pode depender do tamanho q da janela (NGUYEN; WOON; NG, 2015). Se q for muito grande e ocorrer *concept-drift* ou *concept-evolution*, a janela pode conter imagens desatualizadas e a precisão do modelo pode diminuir. Se q for um valor pequeno, a janela pode conter poucas imagens, o que pode ocasionar um ajuste excessivo do modelo. Uma alternativa para esse fato, é considerar um valor q flexível, ou seja, o tamanho da janela muda de acordo com a precisão do modelo. Quando a precisão é alta, o valor de q aumenta. Já quando a precisão é baixa, a janela o valor de q diminui.

Fading Window

Na estratégia *Fading Window*, cada imagem recebe um peso diferente de acordo com seu tempo de chegada, de forma que novas imagens recebam pesos maiores do que as antigas. Com esse método, pode se reduzir a importância de imagens mais antigas e desatualizadas, o que favorece o tratamento de *concept-drift* e *concept-evolution* (NGUYEN; WOON; NG, 2015).

A utilização de janelas de dados favorece a adaptação de algoritmos da classificação *batch* para a utilização na classificação de fluxos de dados. Alguns exemplos de algoritmos de classificação de fluxos de dados são: *Hoeffding Tree* (árvores de decisão), *Bayesian Belief Network* (Naive Bayes) e *Support Vector Machines* (SVM) (NGUYEN; WOON; NG, 2015). Apesar desses algoritmos apresentarem importantes contribuições, para a classificação de fluxos de dados de imagens, é comum se utilizar classificadores baseados em protótipos e classificadores que consideram CNN (NGUYEN; WOON; NG, 2015).

2.3.3.1 *Classificadores baseados em Convolutional Neural Networks*

As CNN *(Convolutional Neural Networks)* são capazes de representar as imagens em vetores de características, conforme definido na Seção 2.1.3. Além disso, as CNN podem ser capazes de realizar a classificação de imagens. Enquanto as camadas iniciais da rede são responsáveis pela representação das imagens, as últimas camadas podem ser responsáveis pela tarefa de classificação (CASTRO et al., 2018).

Além da classificação *batch*, as CNN também podem ser aplicadas para a classificação de fluxos de dados de imagens (REBUFFI et al., 2017; CASTRO et al., 2018). As CNN podem apresentar resultados interessantes em termos de desempenho, mas uma grande vantagem de se utilizar uma única arquitetura de CNN para representação e também para a classificação, seria a criação de uma aplicação fim-a-fim, ou seja, que elimina a

necessidade de exportar a representação dos dados, gerada pela CNN, para a entrada de outro algoritmo de classificação. Alguns exemplos tradicionais de arquiteturas de CNN, que também podem ser aplicados a classificação de fluxos de dados de imagens, são: LeNet, AlexNet, GoogLeNet, DenseNet e VGG16 (GU et al., 2018).

2.3.3.2 *Classificadores baseados em protótipos*

Os algoritmos baseados em protótipos são interessantes para a aplicação em fluxos de dados de imagens, pois, no geral, possuem características incrementais, ou seja, a possibilidade de adicionar ou remover imagens sem a necessidade de reconstruir o modelo, o que pode facilitar o tratamento de *concept-drift* e *concept-evolution*. O algoritmo *k-Nearest Neighbor (KNN)* é um dos algoritmos baseados em protótipos mais famosos para problemas de classificação por conta da sua simplicidade.

Em relação ao algoritmo *KNN* para a classificação *batch*, no contexto de fluxos de dados de imagens o algoritmo *KNN* deve ser alterado para trabalhar com janelas de dados, isto é, apenas uma quantidade de imagens deve permanecer na memória para a atualização do modelo, devido a limitação de memória para lidar com fluxos de dados. Contudo, o tamanho da janela pode influenciar na qualidade dos resultados, por conta da possibilidade de perda de imagens significativas (GARCIA; CARVALHO; MENDES-MOREIRA, 2018; BIFET; HOLMES; PFAHRINGER, 2010) .

Outra estratégia baseada em protótipos, comumente usada para a classificação de fluxos de dados de imagens, é conhecida como NCM (*Neareast Class Mean*) (MENSINK et al., 2013). Essa estratégia considera a utilização de um centróide (μ_j) para representar cada classe de imagens (C_j), conforme a Equação 5:

$$\mu_j = \frac{1}{n_{Cj}} \sum_{x^i \in C_j} x^i \tag{5}$$

Para a atualização online do modelo de decisão do classificador, deve-se recalcular o centróide das classes quando novas imagens rotuladas estão disponíveis no ambiente. Além disso, quando novas classes de imagens surgem, define-se um novo centróide no modelo de decisão para representar a nova classe com base em imagens rotuladas dessa classe. Para a classificação, uma imagem i, representada por um vetor de características $\vec{\boldsymbol{x}}$, é atribuída a uma classe de imagens $c^* \in C$, de acordo com a distância m de $\vec{\boldsymbol{x}}$ em relação ao centróide μ da respectiva classe, conforme a Equação 6:

$$c^* = \underset{c \in C}{\operatorname{argmin}}\, d(\vec{\boldsymbol{x}}, \mu_c) \tag{6}$$

Uma característica dos algoritmos baseados em protótipos é que em ambientes de fluxos de dados, com a ocorrência do fenômeno *concept-drift*, pode ser necessário realizar a atualização dos protótipos utilizados como representantes. Um mecanismo utilizado para realizar essa atualização é esquecer os protótipos que não são adequados para o momento

atual e podem estar desatualizados. Além disso, destaca-se que é comum existirem limitações de memória nesses ambientes. Para tanto, podem ser utilizados métodos de esquecimento de representantes (AGRAHARI; SINGH, 2021), que estão descritos na Seção 2.3.3.3.

2.3.3.3 Métodos de esquecimento de representantes

O trabalho descrito em (AGRAHARI; SINGH, 2021) apresenta as principais técnicas de esquecimento de representantes. Uma das maneiras mais básicas de realizar o esquecimento é considerar aspectos temporais, por exemplo, eliminando representantes mais antigos do modelo de decisão. Ainda sobre aspectos temporais, pode-se utilizar o critério do tempo de requisições, eliminando representantes que o modelo de decisão não utiliza há mais tempo. De forma mais sofisticada, podem ser utilizados critérios de proximidade, eliminando ou unindo representantes que estejam próximos no espaço dimensional. Em cada uma dessas estratégias também é possível considerar o conceito de memória de esquecimento (AGRAHARI; SINGH, 2021). Este conceito implementa um *buffer* de tamanho limitado para inserir temporariamente representantes que foram esquecidos. Métodos que consideram memória de esquecimento acreditam que os representantes ainda podem ser úteis e retornar ao modelo de decisão. A memória de esquecimento visa não eliminar abruptamente representantes que possam contribuir novamente para o modelo de decisão. Porém, caso a memória de esquecimento atinja sua capacidade máxima, os representantes armazenados podem ser eliminados de acordo com algum critério estabelecido, por exemplo, o tempo armazenado.

As principais estratégias de esquecimento de representantes, utilizadas em algoritmos que consideram protótipos (aqui identificados como H), utilizam aspectos temporais (protótipos mais antigos ou mais recentes podem ser eliminados), proximidade e tempo de requisições (AGRAHARI; SINGH, 2021). Essas técnicas estão descritas nos Algoritmos 1, 2, 3, e 4. O Algoritmo 1 considera dois conjuntos de protótipos: H com os protótipos atuais do modelo de decisão e $H^{'}$ com os novos potenciais protótipos. Este método remove os protótipos mais recentes do modelo de decisão (H) e os substitui pelos novos protótipos ($H^{'}$) na mesma quantidade. O objetivo deste método é priorizar as instâncias de dados mais atuais.

Algoritmo 1: Estratégia Mais Recentes

Entrada: $H = (\overrightarrow{\boldsymbol{h}_t}, \overrightarrow{\boldsymbol{h}_{t-1}}, \overrightarrow{\boldsymbol{h}_{t-2}}, ..., , \overrightarrow{\boldsymbol{h}_{t-j}})$ Protótipos atuais do modelo de decisão,
$H^{'} = (\overrightarrow{\boldsymbol{h}_{t'}}, \overrightarrow{\boldsymbol{h}_{t'-1}}, \overrightarrow{\boldsymbol{h}_{t'-2}}, ..., , \overrightarrow{\boldsymbol{h}_{t'-j}})$ Novos protótipos
Saída: $(H$ - $H_f) \cup H'$

```
início
    H_f ← seleciona os |H'| protótipos mais antigos de H;
fim
```

O Algoritmo 2 é mais sofisticado quando comparado ao Algoritmo 1. Este método considera o critério de proximidade para selecionar os protótipos que permanecerão no

modelo de decisão. Primeiramente, os conjuntos de protótipos H e $H^{'}$ são unidos. Em seguida, é calculada a distância entre todos os protótipos. Por fim, os protótipos mais próximos serão analisados, permanecendo somente os mais recentes.

Algoritmo 2: Estratégia Proximidade

Entrada: $H = (\overrightarrow{\boldsymbol{h}_t}, \overrightarrow{\boldsymbol{h}_{t-1}}, \overrightarrow{\boldsymbol{h}_{t-2}}, ..., , \overrightarrow{\boldsymbol{h}_{t-j}})$ Protótipos atuais do modelo de decisão,
$H^{'} = (\overrightarrow{\boldsymbol{h}^{'}_{t'}}, \overrightarrow{\boldsymbol{h}^{'}_{t'-1}}, \overrightarrow{\boldsymbol{h}^{'}_{t'-2}}, ..., , \overrightarrow{\boldsymbol{h}^{'}_{t'-j}})$ Novos protótipos

Saída: H_f - H_p

início
  $H_f \leftarrow H \cup H'$;
  $H_d \leftarrow$ calcula a distância entre todos os protótipos $\in H_f$;
  $H_p \leftarrow$ seleciona os $|H^{'}|$ pares mais próximos de protótipos (considerando H_d) preservando a instância mais recente do par;
fim

O Algoritmo 3 considera o aspecto temporal para selecionar protótipos de forma semelhante ao Algoritmo 1. Porém, este método não considera quando o protótipo foi inserido no modelo de decisão. Considera-se o tempo que o modelo de decisão utilizou o protótipo, ou seja, protótipos mais antigos mas que foram usados recentemente para atribuir instâncias a classes podem ser preservados. O objetivo deste método, em comparação ao Algoritmo 1, é manter os protótipos que foram úteis mais recentemente para o modelo de decisão. O Algoritmo 4 é semelhante ao algoritmo 3, mas introduz o conceito de memória de esquecimento (M).

Algoritmo 3: Estratégia Últimas Requisições

Entrada: $H = (\overrightarrow{\boldsymbol{h}_t}, \overrightarrow{\boldsymbol{h}_{t-1}}, \overrightarrow{\boldsymbol{h}_{t-2}}, ..., , \overrightarrow{\boldsymbol{h}_{t-j}})$ Protótipos atuais do modelo de decisão,
$H^{'} = (\overrightarrow{\boldsymbol{h}^{'}_{t'}}, \overrightarrow{\boldsymbol{h}^{'}_{t'-1}}, \overrightarrow{\boldsymbol{h}^{'}_{t'-2}}, ..., , \overrightarrow{\boldsymbol{h}^{'}_{t'-j}})$ Novos protótipos

Saída: $(H$ - $H_f) \cup H^{'}$

início
  $H_f \leftarrow$ seleciona os $|H^{'}|$ protótipos $\in H$ que não são utilizados a mais tempo;
fim

Algoritmo 4: Estratégia Últimas Requisições com Memória de Esquecimento

Entrada: $H = (\overrightarrow{\boldsymbol{h}_t}, \overrightarrow{\boldsymbol{h}_{t-1}}, \overrightarrow{\boldsymbol{h}_{t-2}}, ..., , \overrightarrow{\boldsymbol{h}_{t-j}})$ Protótipos atuais do modelo de decisão,
$H^{'} = (\overrightarrow{\boldsymbol{h}^{'}_{t'}}, \overrightarrow{\boldsymbol{h}^{'}_{t'-1}}, \overrightarrow{\boldsymbol{h}^{'}_{t'-2}}, ..., , \overrightarrow{\boldsymbol{h}^{'}_{t'-j}})$ Novos protótipos, M Memória de esquecimento

Saída: $(H$ - $H_f) \cup H^{'}$, $M \cup H_f$

início
  $H_f \leftarrow$ seleciona os $|H^{'}|$ protótipos $\in H$ que não são utilizados a mais tempo;
fim

2.4 Avaliação de classificadores de imagens

Os algoritmos para classificação de imagens precisam executar etapas de treinamento e teste. Na tarefa de classificação *batch* é possível dividir o conjunto de imagens em porções, para a realização do treino e teste do classificador (THOMAS; PILLAI, 2019). Já no

cenário de fluxo de dados de imagens, em ambientes de aplicações reais, o conjunto de testes corresponde ao fluxo de imagens. Nesse contexto, devido aos fenômenos *concept-drift* e *concept-evolution*, os classificadores precisam evoluir de maneira online. Essas restrições impõem a necessidade do desenvolvimento de técnicas de classificação e métodos de avaliação específicos para classificadores de fluxos de dados (BIFET et al., 2013). A avaliação de classificadores de imagens é discutida nas Seções 2.4.1 e 2.4.2.

2.4.1 Avaliação *batch*

A avaliação é uma das tarefas fundamentais no processo de classificação, pois ajuda a decidir quais técnicas são mais apropriadas para usar em um problema específico (BIFET; READ, 2018). Para a avaliação da tarefa de classificação *batch*, geralmente, consideram-se técnicas de amostragem para a divisão do conjunto de imagens em porções de teste e treino. Os subconjuntos são separados para assegurar que o desempenho é medido em uma porção de imagens diferente das utilizadas no aprendizado (FACELI; LORENA; GAMA, 2011).

Duas técnicas de amostragem comumente utilizadas para a classificação são: *holdout* e validação cruzada. Na técnica *holdout* considera-se uma porcentagem p de imagens para treinar o classificador e uma porcentagem $(1 - p)$ para o teste. Normalmente, emprega-se $p = 2/3$ (FACELI; LORENA; GAMA, 2011). Uma crítica ao método é que uma combinação diferente de imagens utilizadas para o treino pode influenciar diretamente no desempenho do classificador.

No método de validação cruzada, o conjunto de imagens é divido em r subconjuntos de tamanhos aproximadamente iguais. As imagens de $(r - 1)$ partições são utilizadas no treinamento do classificador. Então, testa-se o modelo de decisão construído na partição restante. Esse processo é repetido r vezes, utilizando em cada ciclo um subconjunto diferente para o teste. O desempenho final equivale a média dos desempenhos observados (WITTEN; FRANK; HALL, 2011).

As técnicas utilizadas para a avaliação da tarefa de classificação *batch* não são adequadas para a classificação de fluxos de dados. Na classificação em ambientes dinâmicos não é possível realizar divisões preliminares do conjunto de dados em porções de treino e teste, pois o treinamento do modelo de decisão acontece várias vezes de maneira online (GAMA, 2010).

2.4.2 Avaliação de fluxos de dados

Para a avaliação dos algoritmos de fluxos de dados é necessário estabelecer métodos para selecionar as instâncias para o teste e também para o treinamento online do classificador. Na literatura científica da área existem abordagens que são comumente usadas

para essa tarefa: *Interleaved Test-Then-Train*, *Prequential* e *Interleaved Chunks* (BIFET; READ, 2018).

A abordagem *Interleaved Test-Then-Train* é uma das mais utilizados para a avaliação em fluxos de dados (BIFET et al., 2015) e está ilustrada na Figura 4. Nesta abordagem, nota-se pela Figura 4, que treina-se o classificador com uma porção de imagens inicial (X^0), então cada imagem que chega no fluxo, representada pelos vetores de características ($\vec{\boldsymbol{x}}^1$, $\vec{\boldsymbol{x}}^2$ ... $\vec{\boldsymbol{x}}^n$), é utilizada para testar o modelo do classificador antes de ser usada para treiná-lo, a partir disso, o modelo pode ser atualizado de forma online. Para tanto, sempre que o rótulo da instância estiver disponível, o modelo de decisão é atualizado. É esperado que, quando surgem instâncias de dados de novas classes no fluxo de dados, o classificador erre a previsão do rótulo dessas instâncias, pois o modelo de decisão ainda não conhece instâncias de dados da nova classe para a atribuição do rótulo correto.

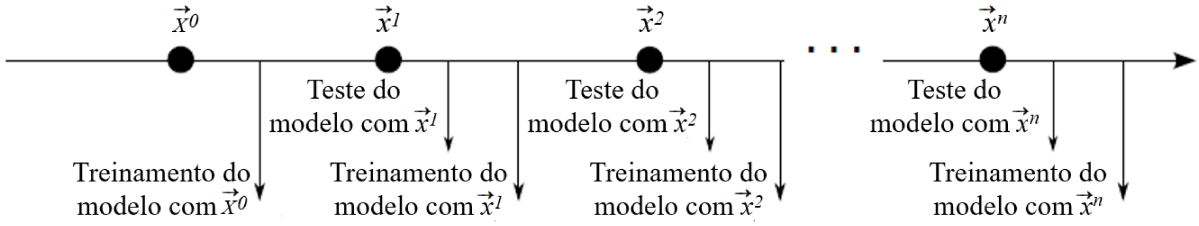

Figura 4 – Abordagem *Interleaved Test-Then-Train*. Adaptado de (STEFANOWSKI; BRZEZINSKI, 2017).

A abordagem de avaliação *Prequential* é semelhante a *Interleaved Test-Then-Train*, pois também se realiza primeiro o teste do classificador e depois o treino online. No entanto, nessa abordagem pode-se utilizar uma *Sliding Window*, definida na Seção 2.3.3, para reunir um número de instâncias para serem avaliadas (GAMA; RODRIGUES; SEBASTIãO, 2009). Com esta abordagem é possível mostrar as mudanças no fluxo de dados mais claramente a cada *Sliding Window* avaliada, o que é importante para detectar as mudanças de conceitos das classes ou o surgimento de novas classes a cada *Sliding Window* (STEFANOWSKI; BRZEZINSKI, 2017).

A Figura 5 é uma variação da abordagem *Interleaved Test-Then-Train* para trabalhar com blocos de imagens, conhecida como *Interleaved Chunks*. Essa abordagem considera blocos de instâncias de dados, de diferentes tamanhos, para testar o modelo de decisão antes de atualizá-lo (ESPAñOL et al., 2015). A partir da Figura 5 é possível notar que nessa abordagem treina-se o classificador com um bloco inicial de imagens (B^0), aguarda-se uma quantidade determinada de imagens que chegam no fluxo para a formação dos blocos (B^1, B^2 ... B^n), que quando estão formados são utilizados para testar o modelo do classificador e depois atualizá-lo. Outra variação da abordagem *Interleaved Test-Then-Train* é chamada de *Delayed Label* (GRZENDA; GOMES; BIFET, 2019). Essa abordagem está ilustrada na Figura 6.

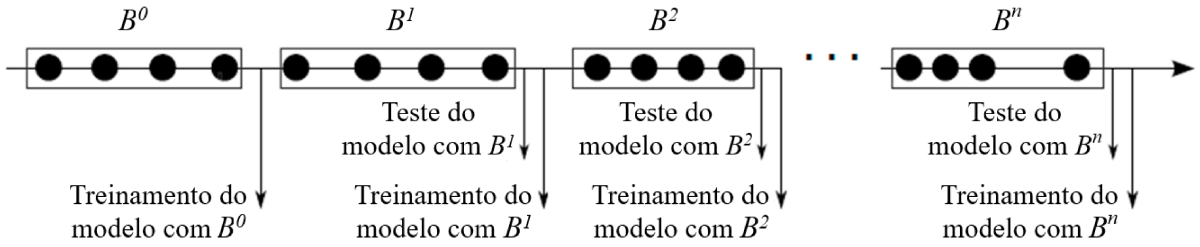

Figura 5 – Abordagem *Interleaved Test-Then-Train* em blocos. Adaptado de (STEFANOWSKI; BRZEZINSKI, 2017).

Na abordagem *Delayed Label* as instâncias de dados são utilizadas para testar e depois para treinar o modelo de maneira online. Entretanto, as instâncias são classificadas em uma classe y com base no modelo de decisão e os rótulos reais $\hat{y}$ dessas instâncias chegam com um certo atraso para o treino. Este atraso é também chamado de latência, conforme descrito na Seção 2.3.1.3. Dessa forma, esta abordagem se aproxima mais de um ambiente real, no qual existe, por exemplo, um usuário especialista para rotular as instâncias em um prazo determinado (GRZENDA; GOMES; BIFET, 2019). Para medir o resultado dos métodos de avaliação utilizados pelos classificadores é necessário considerar medidas de validação. Algumas dessas medidas estão descritas na Seção 2.4.3.

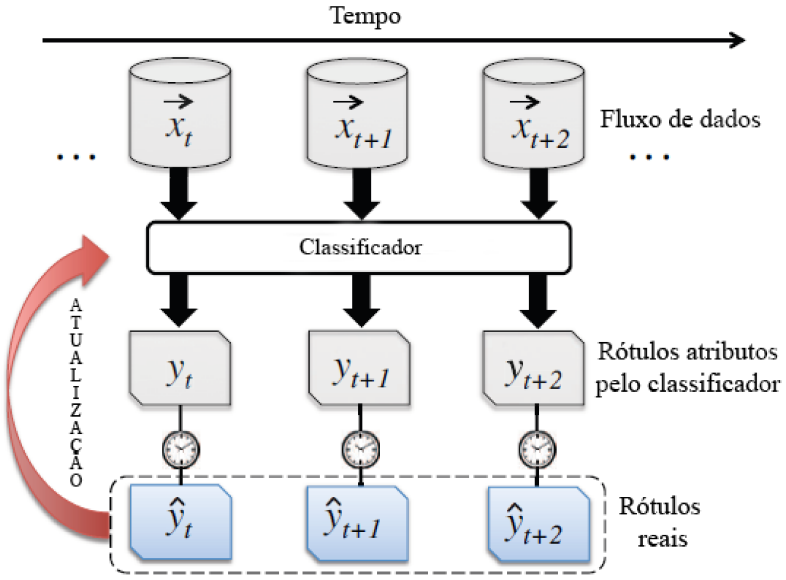

Figura 6 – Abordagem *Delayed Label*. Adaptado de (SOUZA et al., 2015).

2.4.3 Medidas de validação

Na classificação de fluxos de dados de imagens, as medidas acurácia e F-*measure* são, comumente, utilizadas para avaliar o desempenho dos classificadores (BIFET et al., 2015). A acurácia tem a finalidade de observar o desempenho geral do classificador, isto é, a taxa geral de acertos. Contudo, os resultados da acurácia podem ser problemáticos para a análise em cenários com classes desbalanceadas, isto é, com diferentes números de

imagens por classes. Para tanto, a F-*measure* pode complementar a avaliação, pois leva em consideração a questão de classes desbalanceadas (POWERS; AILAB, 2011). A seguir são descritas algumas medidas de validação, que incluem acurácia e F-*measure*. Para a descrição dessas medidas de validação são consideradas as seguintes definições: **TP** - Verdadeiro Positivo; **TN** - Verdadeiro Negativo; **FP** - Falso Positivo; **FN** - Falso Negativo.

- **ACC** (Acurácia): Proporção de predições corretas (N) entre todas as imagens (n) consideradas para a classificação.

$$ACC = \frac{N}{n} \tag{7}$$

- **F1** (*F-measure*): É a média ponderada entre a precisão (Equação 8) e revocação (Equação 9). As fórmulas descritas de precisão e revocação podem considerar a aplicabilidade em problemas de múltiplas classes, como é o caso da classificação de fluxos de dados de imagens.

$$PRE = \frac{1}{n_C} \sum_{c_i \in C} \frac{TP_i}{TP_i + FP_i} \tag{8}$$

$$REV = \frac{1}{n_C} \sum_{c_i \in C} \frac{TP_i}{TP_i + FN_i} \tag{9}$$

$$F1 = 2 * \frac{PRE * REV}{PRE + REV} \tag{10}$$

Além das medidas já mencionadas, outras medidas comumente utilizadas para avaliar o desempenho dos classificadores de fluxos de dados, em contextos diferentes da classificação de imagens, estão descritas em (BIFET et al., 2015), por exemplo, o índice *Kappa*.

2.5 Considerações Finais

Este capítulo apresentou a fundamentação teórica dos principais tópicos abordados nesta tese. Foram descritos os conceitos de representação de imagens, classificação de imagens, métodos de avaliação e técnicas de aprendizado ativo. Sobre os métodos de avaliação, destaca-se que os métodos empregados para a classificação de imagens em cenários estáticos (classificação *batch*) não são adequados para a avaliação em ambientes de fluxos de dados de imagens. Já nos métodos de avaliação de classificadores de fluxos de dados de imagens, é necessário rotular um grande número de instâncias de dados, o que pode inviabilizar a aplicabilidade em cenários reais de aplicações. Contudo, uma alternativa pode ser a utilização de técnicas de aprendizado ativo para a redução do número de instâncias de dados rotuladas. Outra restrição importante para a classificação de fluxos de dados de imagens é a presença de latência para a informação do rótulo

das instâncias de dados. Em alguns cenários de aplicações reais, pode-se considerar um determinado intervalo de tempo (latência intermediária) para a informação do rótulo, mas também é possível que os rótulos nunca sejam informados (latência extrema), o que pode interferir no treinamento online do classificador, pois dificulta a detecção de *concept-drift* e *concept-evolution*. O próximo capítulo tratará dos trabalhos relacionados, que abordam os temas tratados no presente capítulo.

Capítulo 3

Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados que abordam os tópicos descritos no Capítulo 2. A Seção 3.1 descreve os principais trabalhos correlatos que tratam a classificação de fluxos de dados de imagens; a Seção 3.2 discute sobre as características dos métodos de avaliação utilizados nos trabalhos de classificação de fluxos de dados de imagens; a Seção 3.3 descreve sobre técnicas de aprendizado ativo abordadas nos trabalhos relacionados; por fim, a Seção 3.4 apresenta as considerações finais deste capítulo.

3.1 Classificação de fluxos de dados de imagens

Na classificação de fluxos de dados de imagens um dos principais algoritmos utilizados é o NCM (veja Seção 2.3.3). Vários trabalhos correlatos têm considerado a utilização desse algoritmo (HU et al., 2017; RISTIN et al., 2014; HU et al., 2018; REBUFFI et al., 2017). O trabalho descrito em (HU et al., 2017) considerou o algoritmo NCM para a classificação de fluxos de imagens pessoais, já o trabalho (RISTIN et al., 2014) definiu uma abordagem que integrou os algoritmos NCM e *Random Forest* para a classificação de grandes conjuntos de dados. Apesar da estratégia NCM possuir características adequadas para a classificação de fluxos de dados de imagens, os trabalhos analisados consideram suposições em relação aos conjuntos de dados de imagens que podem não ser adequadas para um cenário real, por exemplo: que o rótulo de todas as imagens estão disponíveis ou a existência de um usuário especialista sempre disponível para rotular as imagens, por exemplo, em (HU et al., 2018) todas as novas classes de imagens são informadas por um usuário especialista, existindo forte dependência do usuário. Além disso, o foco desses trabalhos foi avaliar avanços nos classificadores para lidar com os fenômenos *concept-drift* e *concept-evolution*. Entretanto, esses algoritmos não avaliaram sobre *feature-evolution* e latência.

Devido a ocorrência dos fenômenos *concept-evolution* e *concept drift*, no contexto de classificação de fluxos de dados de imagens é importante também considerar a atualização do descritor de características para representar o surgimento de novas características

(*feature-evolution*). O estudo descrito em (REBUFFI et al., 2017) concluiu que o algoritmo NCM não é adequado para a classificação em contextos de *feature-evolution*, devido a dificuldade para a atualização das características do centróide que representa cada classe de imagens. Uma variação do algoritmo NCM, chamada iCaRL (*Incremental Classifier and Representation Learning*), foi proposta com o objetivo de tratar essa questão. O Algoritmo 5 descreve as principais etapas do algoritmo iCaRL.

Algoritmo 5: iCaRL. Adaptado de (REBUFFI et al., 2017)

Entrada: X Conjunto de imagens, r Quantidade de representantes
Saída: eficácia do modelo de classificação f

1 **início**
2 $B \leftarrow$ organização do conjunto de imagens X em *batches* de imagens;
3 $\epsilon \leftarrow$ refinamento do descritor de características CNN com $B[0]$;
4 $W \leftarrow$ extração de características das imagens de $B[0]$ a partir de ϵ;
5 $f \leftarrow$ construção do modelo de decisão inicial a partir de W considerando r representantes para cada classe;
6 **para cada** $B' \in (B - B[0])$ **faça**
7 $Tr \leftarrow$ separação do conjunto de treino a partir de B';
8 $Te \leftarrow$ separação do conjunto de teste a partir de B';
`/* Feature-evolution */`
9 $\epsilon \leftarrow$ refinamento do descritor de características CNN com Tr;
10 $Tr' \leftarrow$ extração de características das imagens de Tr a partir de ϵ;
11 $Te' \leftarrow$ extração de características das imagens de Te a partir de ϵ;
12 $f \leftarrow$ treinamento incremental com Tr' considerando r representantes para cada classe;
13 **para cada** $\vec{\boldsymbol{x}} \in Te'$ **faça**
`/* Atribui o rótulo para a imagem com base no representante mais próximo */`
14 $y_t \leftarrow f(\vec{\boldsymbol{x}})$;
15 **fim**

O algoritmo iCaRL recebe como parâmetros um conjunto de imagens X e uma quantidade de representantes r para representarem cada classe de imagens. Inicialmente, o algoritmo iCaRL realiza uma organização do conjunto X em *batches* de imagens (linha 2 do Algoritmo 5). Na sequência, o primeiro *batch* ($B[0]$), que deve conter todas as imagens rotuladas, é utilizado para o refinamento (*fine-tuning*) (KÄDING et al., 2017)) do descritor de características CNN, que é o descritor ϵ considerado no algoritmo iCaRL para a representação das imagens. Para tanto, foi considerada a arquitetura ResNet. Após o refinamento da CNN, são extraídas as características de $B[0]$ e é realizado o treinamento inicial do classificador, considerando a quantidade de r representantes para cada classe de imagens existente em $B[0]$. Para a seleção dos representantes, é calculado um centróide de cada classe. São selecionadas as instâncias de dados mais próximas desse centróide para formarem o conjunto de representantes da respectiva classe.

Na linha 6 do Algoritmo 5 inicia-se o processamento dos *batches* de imagens, com exceção de $B[0]$, que já foi considerado para o treinamento inicial de f. Para cada *batch* é realizada uma divisão das imagens entre porções de treino (Tr) e teste (Te). A porção de treino deve conter todas as imagens rotuladas. Dessa forma, ocorre a etapa de *feature-evolution* na linha 9 do Algoritmo 5. Para tanto, utiliza-se as imagens do conjunto Tr e recursos de *backpropagation* para a atualização de pesos da CNN e, consequentemente, para melhorar a capacidade de representação das imagens com as novas imagens disponíveis. A representação das imagens já processadas, mas que foram armazenadas como representantes, também é atualizada automaticamente com esse processo.

Na linha 12 do Algoritmo 5 ocorre o treinamento incremental de f com as imagens de Tr. Nessa etapa, considera-se também o limite de r representantes para cada classe de imagens. Caso sejam adicionados novos representantes em uma classe já conhecida, utiliza-se um recurso de esquecimento que remove os representantes mais antigos e prioriza as novas instâncias de dados. Por fim, cada imagem do conjunto Te é avaliada pelo modelo de decisão f para a atribuição de um rótulo y

A partir da descrição do Algoritmo 5, percebe-se que o algoritmo iCaRL considera importantes recursos para a classificação de fluxos de dados de imagens, por exemplo, a utilização de múltiplos representantes para cada classe de imagens. Se comparado ao algoritmo NCM, que considera apenas um representante, essa característica do algoritmo iCaRL é uma vantagem. Além disso, essa estratégia permite a aplicação do recurso de *feature-evolution*, pois possibilita a atualização dos representantes já conhecidos pelo modelo. Com esses recursos e com os experimentos descritos em (REBUFFI et al., 2017) é possível constatar que o algoritmo iCaRL pode apresentar eficácia superior ao algoritmo NCM. Entretanto, apesar das importantes características do algoritmo iCaRL, essa abordagem não trata de fato a ocorrência do fenômeno *concept-evolution*, pois as novas classes de imagens são utilizadas para o aprendizado incremental do modelo de decisão, mas o teste do modelo de decisão sempre é realizado com as mesmas classes que já foram utilizadas no aprendizado (treino), isto é, novas classes nunca são apresentadas diretamente para o teste do modelo de decisão.

Além do trabalho (REBUFFI et al., 2017), a questão do *feature-evolution* também foi abordada em (CASTRO et al., 2018). O trabalho apresentando em (CASTRO et al., 2018) propõe uma única arquitetura de CNN, denominada *fim-a-fim*, para a representação das imagens e para a classificação, isto é, a mesma CNN foi utilizada tanto para a representação e atualização dos descritores de características quanto para a classificação das imagens. O objetivo dessa estratégia é eliminar a necessidade de possuir dois ambientes: um para a representação das imagens e outro para a classificação. Em termos de eficácia, essa estratégia foi comparada ao algoritmo iCaRL, e apresentou resultados interessantes, inclusive superiores em alguns conjuntos de dados. Apesar das importantes contribuições dos trabalhos realizados em (REBUFFI et al., 2017; CASTRO et al., 2018) sobre o recurso

de *feature-evolution*, não foi avaliada a influência da quantidade de imagens e de classes de imagens utilizadas para o refinamento do descritor de características CNN. Em (REBUFFI et al., 2017) foi avaliada somente a redução do número de classes na representação das imagens, mas não foram realizadas variações no número de imagens de cada classe. O trabalho descrito em (NAKATA et al., 2022) foi inspirado no algoritmo iCaRL, mas o foco da investigação foi desenvolver mecanismos para o armazenamento de alta capacidade e não foram investigadas outras características desse contexto, como a ocorrência do fenômeno *concept-evolution*. A abordagem proposta em (ZHENG; WEN, 2022) também desenvolveu uma única arquitetura de CNN para a representação das imagens e para a classificação. Em (ZHENG; WEN, 2022) foi considerada a ocorrência do fenômeno *concept-evolution*, mas o recurso de *feature-evolution* não foi explorado.

A complexidade computacional do algoritmo iCaRL pode ser representada como: $O((g * e * b^2)) + (b^2))$, onde $(g * e * b^2))$ representa a etapa de descrição de características das imagens e também o recurso de *feature-evolution*. O valor e representa o número de épocas considerado na CNN, o valor b representa o número de *batches* de imagens e o valor g corresponde ao número de vezes que a rede passará pelo processo de *feature-evolution*. O cálculo de (b^2) representa o processamento de cada *batch* para a etapa de classificação.

O trabalho descrito em (WANG et al., 2019) também apresenta uma proposta para a classificação de fluxos de dados de imagens, mas que considera a questão da alta dimensionalidade dos dados. Em (REBUFFI et al., 2017; CASTRO et al., 2018; ZHENG; WEN, 2022) essa questão não foi abordada. Os autores de (WANG et al., 2019) afirmam que a estratégia NCM e outras estratégias de classificação de fluxo de dados (HAQUE; KHAN; BARON, 2016; HAQUE et al., 2016; MU et al., 2017; MASUD et al., 2013) podem ter o desempenho degradado em conjuntos de dados de imagens, pois tais estratégias consideram cálculos de distâncias para a determinação das classes das instâncias de dados. Geralmente, as imagens são representadas por vetores de alta dimensão (veja Seção 2.1). Portanto, os classificadores podem sofrer com a maldição da dimensionalidade em fluxos de dados de imagens. Dessa forma, foi proposto o *framework* CPE (*CNN-based Prototype Ensemble*) em (WANG et al., 2019) para a classificação de conjuntos de fluxos de dados de alta dimensão.

Em (WANG et al., 2019) o problema da alta dimensionalidade dos dados foi abordado, por meio da utilização de uma CNN capaz de gerar vetores de características com um número reduzido de atributos. Nessa CNN, a representação das imagens é continuamente atualizada, isto é, quando novas instâncias de dados rotuladas se tornam disponíveis no modelo de decisão, a representação das imagens também sofre um processo de *feature-evolution*. Dessa forma, o processo de redução da dimensionalidade também é realizado durante todo o processamento do fluxo de dados de imagem, já que novas classes de imagens podem surgir e, consequentemente, diferentes características das imagens devem ser consideradas para a redução da dimensionalidade. Outra característica do

algoritmo CPE é que múltiplos protótipos (representantes) são considerados na fase de classificação para representar as classes de imagens. O Algoritmo 6 apresenta uma visão geral do algoritmo CPE.

O Algoritmo 6 recebe como parâmetro um conjunto de imagens rotulado X, que é utilizado para o treinamento inicial do modelo de decisão. Além disso, são informados: o fluxo de dados S; um valor r que define a quantidade máxima de representantes; um limiar L que determina uma distância máxima para a classificação; um limiar N que define que uma instância de dados pertence a uma provável nova classe; um limiar O que determina a proximidade entre dois representantes; e um valor T que determina o tamanho máximo do *buffer* considerado no algoritmo CPE.

O primeiro passo do algoritmo CPE é o refinamento do descritor de características CNN (ϵ) com o conjunto de dados inicial X (linha 2 do Algoritmo CPE). Na sequência, a partir de ϵ são extraídas as características das imagens de X. Então, o conjunto X é organizado de acordo com cada classe de imagens, resultando no conjunto X'. Na linha 4 do Algoritmo 6 é realizado o trabalho de redução da dimensionalidade. Para tanto, o algoritmo CPE possui um recurso que, a partir da separação por classes (X'), é capaz de identificar as características mais significativas para a distinção entre as classes. Dessa forma, os vetores de características, responsáveis por descrever cada imagem, são compostos somente pelas características que foram identificadas com esse mecanismo, promovendo a redução da dimensionalidade em relação ao tamanho original do vetor produzido por ϵ. Após essa etapa, é realizada a seleção dos representantes de cada classe. São selecionadas as r instâncias de dados mais próximas do centróide de cada classe de imagem. Os representantes selecionados são considerados para a construção do modelo de decisão f.

A partir da linha 7 do Algoritmo 6 inicia-se a fase *online* do algoritmo, com o processamento do fluxo de dados S. Cada vetor de características $\vec{\boldsymbol{x}}$ *in* S é avaliada por f. Caso a distância de $\vec{\boldsymbol{x}}$ para o representante mais próximo seja igual ou inferior ao limiar L, o rótulo do respectivo representante é atribuído a $\vec{\boldsymbol{x}}$ (linha 10 ao Algoritmo 6). Caso contrário, $\vec{\boldsymbol{x}}$ é adicionado ao *buffer* Z (linha 12 ao Algoritmo 6). Quando o tamanho do *buffer* Z atinge o valor definido pelo parâmetro T, ocorre uma fase de aprendizado ativo. Para tanto, são selecionadas as instâncias de dados que a distância para o representante mais próximo é superior ao limiar N. Esse limiar tem o objetivo de determinar quais instâncias de dados provavelmente pertencem a novas classes. Essas instâncias de dados são armazenadas no conjunto R (linha 14 do Algoritmo 6). Na sequência, são solicitados os rótulos de R. Então, o conjunto de representantes de classes é atualizado com as novas instâncias de dados rotuladas (linha 16 do Algoritmo 6).

Com o processo de aprendizado ativo do algoritmo CPE, é possível realizar a atualização do descritor de características ϵ, a partir do refinamento da CNN com as novas instâncias de dados rotuladas (linha 17 do Algoritmo 6). Na sequência, considerando a

Algoritmo 6: CPE. Adaptado de (WANG et al., 2019)

Entrada: X conjunto de imagens rotulado, S fluxo de imagens, r Quantidade de representantes, L limiar de distância, N limiar de distância para novidade, O limiar de distância para representantes, T tamanho do *buffer*
Saída: eficácia do modelo de classificação f

1 **início**
  /* Fase offline */
2 $\epsilon \leftarrow$ refinamento da CNN com X;
3 $X' \leftarrow$ extração das características das imagens de X com ϵ e separação das instâncias de X de acordo com a classe;
4 $X'' \leftarrow$ redução da dimensionalidade de X' e seleção dos representantes considerando r;
5 $f \leftarrow$ construção do modelo de decisão inicial a partir de X'';
  /* Fase online */
6 $Z \leftarrow \emptyset$;
7 **repita**
8 recebe $\vec{\boldsymbol{x}}$ de S extraindo as características com ϵ;
  /* $\vec{\boldsymbol{x}}$ é avaliado por f */
9 **se** *a distância de* $\vec{\boldsymbol{x}}$ *em relação ao representante* $\vec{\boldsymbol{x'}} \in X''$ *mais próximo* $\leq L$ **então**
10 y_t de $\vec{\boldsymbol{x}} \leftarrow y_t$ de $\vec{\boldsymbol{x'}}$;
11 **senão**
12 $Z \leftarrow Z \cup \vec{\boldsymbol{x}}$;
13 **se** $|Z| == T$ **então**
14 $R \leftarrow$ identificação das instâncias de Z a serem rotuladas considerando as distâncias para $\vec{\boldsymbol{x'}} \in X''$ e o limiar N;
15 $R' \leftarrow$ solicitação dos rótulos $\hat{y}$ de R;
16 $X'' \leftarrow X'' \cup R'$;
  /* *Feature-evolution* */
17 $\epsilon \leftarrow$ refinamento do descritor de características CNN os novos representantes;
  /* atualiza incrementalmente o modelo de decisão a partir de X'' */
18 $X'' \leftarrow$ redução da dimensionalidade de X'' e redução do número de representantes considerando r e O;
19 **até** *o fim de* S ;
20 **fim**

potencial ocorrência dos fenômenos *concept-drift* e *concept-evolution* e que a representação dos vetores de características das imagens foi atualizada, é realizada a redução da dimensionalidade dos representantes, da mesma forma que foi realizada na linha 4 do Algoritmo 6. Por fim, também é realizado um processo de esquecimento de representantes. Para tanto, observa-se a existência de representantes próximos em relação a um limiar O. Caso existam, esses representantes passam por uma etapa de fusão, de forma que dois ou mais representantes se tornem apenas um no modelo de decisão.

Os resultados dos experimentos com o algoritmo CPE, publicados em (WANG et al., 2019), são promissores, uma vez que consideram importantes aspectos da classificação

de fluxos de dados de imagens, tais como: ocorrência de *concept-drift* e *concept-evolution*, aprendizado ativo e a alta dimensionalidade dos dados. Entretanto, os experimentos de (WANG et al., 2019) não compararam o algoritmo CPE com outras técnicas que tratam a alta dimensionalidade dos dados. Além disso, também não foram consideradas outras estratégias de aprendizado ativo. Outro ponto é que o algoritmo CPE possui vários parâmetros, por exemplo, os valores r, L, N, O e T considerados no Algoritmo 6. Essa característica pode dificultar a encontrar o melhor conjunto de parâmetros para conjuntos de dados que não foram considerados na experimentação de (WANG et al., 2019) e na aplicabilidade em cenários de aplicações reais.

Em relação a complexidade computacional, no algoritmo CPE a utilização da CNN para a descrição de características e para o tratamento da alta dimensionalidade dos dados é um dos pontos mais relevantes. Considerando C uma constante para operações de otimização, c a quantidade de classes conhecidas pelo modelo, e o número de épocas e b o número de *batches* de imagens, a complexidade computacional do algoritmo CPE pode ser representada como: $O(C * e * (b^2 + |Z| * c))$ (WANG et al., 2019).

3.2 Avaliação de classificadores em fluxos de dados

O método de avaliação considerado no trabalho descrito em (WANG et al., 2019) considera aspectos interessantes para a classificação de fluxos de dados de imagens, por exemplo: considera o treinamento inicial do modelo de decisão com um conjunto de imagens rotuladas, é capaz de considerar a ocorrência do fenômeno *concept-evolution* e também considera a classificação com atraso (latência) por meio da utilização de um *buffer* de imagens. Entretanto, em (WANG et al., 2019) não fica evidente como o fenômeno *concept-drift* foi considerado no método de avaliação, já que as imagens foram selecionadas em ordens aleatórias. Além disso, foi considerado o mesmo tamanho de *buffer* para todos os conjuntos de dados considerados. Já os trabalhos apresentados em (REBUFFI et al., 2017; CASTRO et al., 2018), apesar de considerarem a natureza dinâmica do problema no qual o modelo de decisão precisa ser constantemente atualizado, avaliam tais modelos em cenários controlados, por exemplo, sem a ocorrência do fenômeno *concept-evolution*, o que pode ser incompatível com as características de fluxos de dados.

O método de avaliação utilizado nos trabalhos (REBUFFI et al., 2017; CASTRO et al., 2018) divide o conjunto de imagens em *batches* de classes. Por exemplo, dado um conjunto de imagens que possui c classes de imagens, divide-se o conjunto em n *batches* de m classes cada, de forma que $n \times m = c$. Cada *batch* é disponibilizado, sequencialmente, para o treinamento incremental do classificador. Entre cada etapa de treino, é realizado o respectivo teste com as classes disponíveis até então. O resultado do classificador corresponde a média dos desempenhos computados em cada *batch*. Esse método de avaliação está ilustrado na Figura 7.

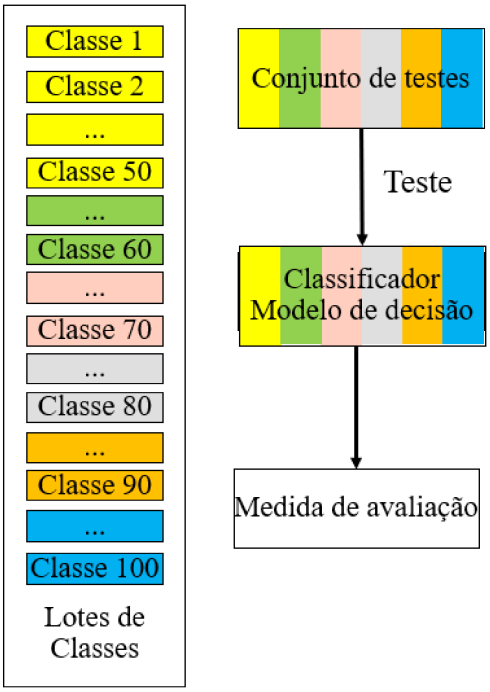

Figura 7 – Ilustração do método de avaliação utilizado nos trabalhos (REBUFFI et al., 2017; CASTRO et al., 2018).

Conforme ilustrado na Figura 7, os *batches* são compostos por diferentes classes de imagens e estão ilustrados por cores distintas na Figura 7. Observa-se também, que esse método de avaliação não considera que novas classes de imagens podem surgir para testar o classificador antes de treiná-lo. Nesse método as imagens sempre são apresentadas primeiramente para o treino e depois para o teste, o que não permite a ocorrência de *concept-evolution*. Além disso, esses trabalhos supõe um conjunto de imagens rotuladas, inclusive com exemplos de novas classes, para a atualização do modelo. Essa suposição pode não refletir cenários de aplicações reais, nos quais diferentes classes de imagens chegam continuamente e, geralmente, não se dispõe de um especialista para rotular todas as imagens. Dessa forma, observa-se que uma das principais características consideradas em métodos de avaliação, inadequados para cenários de aplicações reais, é relativa a atualização do modelo de decisão. Para tanto, um recurso que pode ser utilizado para auxiliar nessa tarefa é utilização de técnicas de aprendizado ativo, que estão descritas na Seção 3.3.

3.3 Aprendizado ativo

Exemplos de trabalhos correlatos que lidam com a questão de indisponibilidade de rótulos para a atualização do modelo de decisão são descritos em (PARREIRA; PRATI, 2019; SOUZA et al., 2015). Esses trabalhos consideram atrasos, denominados latência (veja Seção 2.3.1.3), para a chegada dos rótulos das instâncias. O uso de latência permite

aos métodos a simulação de ambientes de aplicações reais, nos quais não se tem sempre a disposição, por exemplo, um usuário especialista para rotular as imagens. Alguns desafios para o emprego de latência na classificação de fluxo de dados de imagens são relacionados a taxa de atraso considerada e a disponibilidade do usuário especialista, pois são fatores que podem impactar a qualidade da classificação.

O trabalho definido em (PARREIRA; PRATI, 2019), além de considerar latência, descreve técnicas de aprendizado ativo que podem ser utilizadas na classificação de fluxos de dados. As estratégias de (PARREIRA; PRATI, 2019) foram baseadas nos métodos descritos em (ŽLIOBAITė et al., 2014) (veja Seção 2.3.2). Em (PARREIRA; PRATI, 2019) também é apresentado um método capaz de calcular o custo para rotular cada instância. Para a aplicação das técnicas de aprendizado ativo propostas, foi definido em (PARREIRA; PRATI, 2019) o *framework* descrito no Algoritmo 7. Esse *framework* considera a presença de um oráculo para a informação dos rótulos, que em cenários de aplicações reais poderia ser um usuário especialista.

O *framework* de (PARREIRA; PRATI, 2019) considera como parâmetros: o fluxo de dados S, um modelo de decisão f para a classificação das instâncias de dados e uma estratégia de aprendizado ativo E. Cada instância de dados $\vec{\boldsymbol{x}} \in S$ é avaliado por f recebendo um rótulo y (linha 4 do Algoritmo 7). Essa mesma instância de dados é inserida em um *buffer* Z. Na sequência inicia-se o processo de aprendizado ativo. Primeiramente, é verificado se o oráculo está apto para informar os rótulos (linha 7 do Algoritmo 7). Para tanto, são considerados três fatores: latência, capacidade máxima de rótulos e disponibilidade. Deve ser verificado se o atraso (latência) considerado já foi atendido. Também se considera que o oráculo pode ter atingido sua capacidade máxima de informar rótulos, esse recurso é comumente conhecido como *budge*. Por fim, o oráculo também pode estar indisponível rotulando outras instâncias de dados naquele momento.

Algoritmo 7: *Framework* definido em (PARREIRA; PRATI, 2019)

Entrada: S fluxo de dados, f modelo de classificação, E Estratégia de aprendizado ativo
Saída: predições de f

```
1  início
2      Z ← ∅;
3      repita
4          recebe x⃗ de S;
5          y_t ← f(x⃗);
6          Z ← Z ∪ x⃗;
7          se oráculo está disponível então
8              x⃗′ ← seleciona uma instância de dados a ser rotulada considerando E;
9              ŷ ← recebe o rótulo verdadeiro de x⃗′;
10             atualiza incrementalmente f com ŷ;
11             Z ← Z - x⃗′;
12     até o fim de S ;
13 fim
```

No Algoritmo 7, caso o oráculo estiver disponível, seleciona-se uma instância de dados $\vec{\boldsymbol{x}}'$, que terá o rótulo verdadeiro ($\hat{y}$) informado. Então, pode-se realizar o treinamento online do modelo de decisão f com a nova instância de dados rotulada (linha 10 do Algoritmo 7). Para selecionar a instância de dados $\vec{\boldsymbol{x}}'$ é necessário considerar uma estratégia de aprendizado ativo E. Para tanto, foram descritas em (PARREIRA; PRATI, 2019) as estratégias de aprendizado ativo detalhadas nos Algoritmos 8, 9, 10, 11, 12 e 13.

Os Algoritmos 8 e 9 descrevem estratégias de aprendizado ativo que consideram aspectos temporais. No Algoritmo 8 é selecionada para rotular a instância de dados mais recente ($\vec{\boldsymbol{x}_t}$) do *buffer* Z. Já no Algoritmo 9 é selecionada a instância de dados menos recente, isto é, a que está a mais tempo ($\overrightarrow{\boldsymbol{x}_{t-j}}$) no *buffer* Z.

Algoritmo 8: Estratégia Mais Recente. Adaptado de (PARREIRA; PRATI, 2019)

Entrada: $Z = (\vec{\boldsymbol{x}_t}, \overrightarrow{\boldsymbol{x}_{t-1}}, \overrightarrow{\boldsymbol{x}_{t-2}}, ..., , \overrightarrow{\boldsymbol{x}_{t-j}})$ *Buffer* de instâncias de dados
Saída: $\vec{\boldsymbol{x}_t}$

Algoritmo 9: Estratégia Menos Recente. Adaptado de (PARREIRA; PRATI, 2019)

Entrada: $Z = (\vec{\boldsymbol{x}_t}, \overrightarrow{\boldsymbol{x}_{t-1}}, \overrightarrow{\boldsymbol{x}_{t-2}}, ..., , \overrightarrow{\boldsymbol{x}_{t-j}})$ *Buffer* de instâncias de dados
Saída: $\overrightarrow{\boldsymbol{x}_{t-j}}$

O Algoritmo 10 descreve uma das estratégias mais simples de seleção de instâncias para a informação dos rótulos. Nessa estratégia seleciona-se de maneira aleatória a instância que terá o rótulo informado. Para tanto, utiliza-se uma função δ que retorna um valor a aleatório no intervalo $[t, t-j]$. Então, a instância de dados na respectiva posição a é selecionada ($\vec{\boldsymbol{x}_a}$).

Algoritmo 10: Estratégia Aleatória. Adaptado de (PARREIRA; PRATI, 2019)

Entrada: $Z = (\vec{\boldsymbol{x}_t}, \overrightarrow{\boldsymbol{x}_{t-1}}, \overrightarrow{\boldsymbol{x}_{t-2}}, ..., , \overrightarrow{\boldsymbol{x}_{t-j}})$ *Buffer* de instâncias de dados
Saída: $\vec{\boldsymbol{x}_a}$
1 **início**
2 $\quad a \leftarrow \delta[t, t-j]$;
3 **fim**

Apesar da capacidade de seleção de instâncias de dados, as estratégias de aprendizado ativo descritas nos Algoritmos 8, 9 e 10 não consideram nenhum mecanismo que considere o grau de representatividade das instâncias de dados. Dessa forma, no Algoritmo 11 é detalhada a estratégia baseada em incerteza. Nessa estratégia considera-se uma função φ que determina as probabilidades das instâncias de dados pertencerem as respectivas classes. Então, seleciona-se a instância de dados com o maior grau de incerteza, isto é, com as menores probabilidades detectadas.

Algoritmo 11: Estratégia Incerteza. Adaptado de (PARREIRA; PRATI, 2019)

Entrada: $Z = (\overrightarrow{\boldsymbol{x}_t}, \overrightarrow{\boldsymbol{x}_{t-1}}, \overrightarrow{\boldsymbol{x}_{t-2}}, ..., , \overrightarrow{\boldsymbol{x}_{t-j}})$ *Buffer* de instâncias de dados
Saída: $\overrightarrow{\boldsymbol{x}_p}$
início
  $P \leftarrow \varphi(Z)$;
  $\overrightarrow{\boldsymbol{x}_p} \leftarrow \operatorname{argmin} P$;
fim

Na estratégia baseada em incerteza, descrita no Algoritmo 11, é considerado o grau de informatividade da instância de dados, mas não se leva em consideração o esforço necessário para a obtenção do rótulo. Algumas instâncias de dados podem demandar maior esforço para a interpretação do oráculo, por exemplo, quando possuem probabilidades semelhantes de pertencerem a duas ou mais classes. Nessa situação, o oráculo pode ficar indisponível por um longo intervalo de tempo, o que pode interferir na capacidade de informação de rótulos de outras instâncias de dados. Dessa forma, a estratégia de aprendizado ativo descrita no Algoritmo 12 considera, além da incerteza, o custo para a informação do rótulo. Para tanto, utiliza-se uma função ζ que multiplica a probabilidade de pertencer as classes pelo custo detectado. Dessa forma, é seleciona a instância de dados que apresente o melhor valor considerando a relação incerteza e custo.

Algoritmo 12: Estratégia Incerteza com Custo. Adaptado de (PARREIRA; PRATI, 2019)

Entrada: $Z = (\overrightarrow{\boldsymbol{x}_t}, \overrightarrow{\boldsymbol{x}_{t-1}}, \overrightarrow{\boldsymbol{x}_{t-2}}, ..., , \overrightarrow{\boldsymbol{x}_{t-j}})$ *Buffer* de instâncias de dados, Variável v
Saída: $\overrightarrow{\boldsymbol{x}_{pc}}$
início
  $P_c \leftarrow \zeta(Z)$;
  $\overrightarrow{\boldsymbol{x}_{pc}} \leftarrow \operatorname{argmin} P_c$;
fim

Na estratégia de aprendizado ativo descrita no Algoritmo 12 foi considerado o grau de informatividade e também o custo para a obtenção do rótulo. Entretanto, podem ocorrer casos que alguma das classes apresenta custos menores para obtenção de rótulos em relação a outras classes. Dessa forma, essa estratégia pode enviesar a seleção de instâncias de dados, priorizando as classes com custo reduzido. De forma a tentar evitar essa situação, a estratégia de aprendizado ativo descrita no Algoritmo 13 considera também o aspecto aleatório para a seleção da instância de dados para a obtenção do rótulo. Para tanto, utiliza-se um valor v como parâmetro, que será considerado por uma função δ. Essa função retorna um valor aleatório a no intervalo $[0, v]$. Se o valor de a é igual a 0, a estratégia aleatória (Algoritmo 10) é utilizada para a seleção da instância. Senão, a estratégia de incerteza com custo é considerada (Algoritmo 12). Com isso, observa-se que o valor v é importante para determinar qual estratégia será considerada. Com valores superiores de v a tendência é que a estratégia de incerteza com custo seja executada mais vezes em

relação a estratégia aleatória.

Algoritmo 13: Estratégia Incerteza com Custo e Aleatória. Adaptado de (PARREIRA; PRATI, 2019)

Entrada: $Z = (\overrightarrow{\boldsymbol{x}_t}, \overrightarrow{\boldsymbol{x}_{t-1}}, \overrightarrow{\boldsymbol{x}_{t-2}}, ..., , \overrightarrow{\boldsymbol{x}_{t-j}})$ *Buffer* de instâncias de dados
Saída: $\overrightarrow{\boldsymbol{x}_i}$

```
início
    a = δ[0, v];
    se a==0 então
        x_i ← Aleatoria(Z);
    senão
        x_i ← IncertezaCusto(Z);
fim
```

Outra estratégia de aprendizado ativo aplicada a classificação de fluxos de dados foi descrita em (CAVALCANTI, 2021). Essa estratégia é baseada na abordagem *query-by-committee* (veja Seção 2.3.2). O trabalho de (CAVALCANTI, 2021) considerou a aplicação dessa estratégia em classificadores baseados em agrupamento de dados. Os resultados dessa estratégia mostram que os classificadores apresentaram maiores valores de eficácia quando comparados a abordagem original dos algoritmos sem a estratégia de aprendizado ativo.

A partir das estratégias de (PARREIRA; PRATI, 2019) e (CAVALCANTI, 2021) observa-se a diversidade de critérios que podem ser utilizados em estratégias de aprendizado ativo para a seleção de instâncias de dados. Nota-se também a partir de (PARREIRA; PRATI, 2019) a possibilidade de combinar critérios nessas estratégias. Entretanto, as estratégias de aprendizado ativo não foram utilizadas com conjuntos de dados de imagens em (PARREIRA; PRATI, 2019; CAVALCANTI, 2021). Geralmente, uma característica de conjuntos de dados de imagens é alta dimensionalidade dos dados (veja Seção 2.1). As técnicas de (PARREIRA; PRATI, 2019) que utilizam funções de distância podem ter o desempenho degradado nesse contexto. Além disso, as estratégias de (CAVALCANTI, 2021) foram aplicadas somente a algoritmos baseados em agrupamento de dados. Portanto, é necessário investigar se essas estratégias podem ser aplicadas a classificadores de fluxos de dados de imagens e analisar estratégias que podem ser estendidas para o uso nessa situação.

3.4 Considerações Finais

Este capítulo apresentou uma visão geral sobre os principais trabalhos relacionados aos tópicos abordados nesta monografia. Foram apresentados os principais algoritmos utilizados para a classificação de fluxos de dados de imagens. Além disso, discutiu-se sobre os respectivos métodos de avaliação empregados nesses algoritmos. Observou-se que

os métodos de avaliação utilizados podem considerar restrições que não são compatíveis com cenários de aplicações reais, por exemplo, possuir um grande número de imagens rotuladas e não levar em consideração os fenômenos *concept-evolution* e *concept-drift*. Por fim, trabalhos de aprendizado ativo foram discutidos. Observou-se que as técnicas de aprendizado ativo podem contribuir para a seleção de instâncias a serem rotuladas em fluxos de dados. Além disso, destaca-se que algumas dessas técnicas consideram a presença de latência, que é uma característica importante em fluxos de dados. Entretanto, essas técnicas ainda não foram utilizadas em trabalhos de classificação de fluxos de dados de imagens e é necessário avaliar o comportamento em ambientes de alta dimensionalidade.

Capítulo 4

Proposta para a classificação de fluxos de dados de imagens

A tarefa de classificação de imagens possui várias etapas, que abrangem desde a aquisição das imagens até a avaliação do classificador. Este capítulo apresenta cada etapa do método de pesquisa considerado neste trabalho para a classificação de fluxos de dados de imagens. Além disso, é detalhado como cada etapa foi explorada no trabalho desenvolvido, evidenciando a organização dos próximos capítulos da tese. A Seção 4.1 apresenta a formalização do problema de classificação de imagens, a Seção 4.2 descreve as etapas para a classificação de fluxos de dados de imagens. A Seção 4.3 descreve como as etapas da classificação de fluxos de dados de imagens foram exploradas no trabalho desenvolvido. Por fim, a Seção 4.4 apresenta as considerações finais do capítulo.

4.1 Formalização do problema

A representação de conjuntos de imagens empregada em tarefas de mineração de dados, como a classificação, é normalmente baseada em características extraídas de cada imagem (veja Seção 2.1). Assim, o processo de extração de características de uma dada imagem i gera um vetor $\vec{\boldsymbol{x}}$ de d dimensões para representar a imagem i, onde cada dimensão de $\vec{\boldsymbol{x}}$ corresponde a uma característica da imagem i obtida pelo método de extração de características adotado.

Em termos gerais, dado um conjunto rotulado de imagens, representadas por vetores de características $\vec{\boldsymbol{x}}$, a classificação de imagens prediz a classe de uma nova imagem a partir de um modelo de decisão f criado usando as imagens rotuladas. Formalmente: dado um conjunto de imagens da forma $(\vec{\boldsymbol{x}},\ y)$, tal que $\vec{\boldsymbol{x}} = x_1, ..., x_d$ é um vetor de características que descreve uma imagem e y é uma classe discreta de um conjunto C com n_C diferentes classes, o classificador cria um modelo $y = f(\vec{\boldsymbol{x}})$ para prever as classes y de outras imagens.

Em cenários de aplicações reais, considera-se a existência de fluxos de dados de imagens. Formalmente, um fluxo de dados de imagens S, é uma sequência de imagens i^1, i^2, ..., i^n, potencialmente infinita, representadas por vetores de características $\vec{\boldsymbol{x}}$, que chega nos *timestamps* t^1, t^2, ... ,t^n. Para a classificação de fluxos de dados de imagens pode-se considerar um conjunto inicial de imagens rotuladas X para a criação de um modelo $f(\vec{\boldsymbol{x}})$ para prever as classes $y \in \{c^1, c^2, ..., c^{n_C}\}$ de novas imagens que chegam no fluxo de imagens S. Entretanto, devido aos fenômenos *concept-drift* e *concept-evolution* (veja Seção 2.3.1), nesse contexto o modelo de decisão $f(\vec{\boldsymbol{x}})$ não pode ser treinado uma única vez, já que no treinamento inicial foram consideradas n_C classes e no teste do modelo de decisão podem surgir w novas classes. Dessa forma, deve-se treinar o modelo $f(\vec{\boldsymbol{x}})$ de forma online para que ele seja capaz de atribuir um rótulo y' a um vetor de características $\vec{\boldsymbol{x}}$, sendo $y' \in \{c^1, c^2, ..., c^{n_C}, c^{(n_C)+1}, ..., c^{(n_C)+w}\}$.

Para verificar o desempenho do modelo de decisão $f(\vec{\boldsymbol{x}})$, deve-se utilizar algum método de avaliação. Para a avaliação da tarefa de classificação *batch* de imagens (cenários estáticos), geralmente, divide-se um conjunto X de imagens em duas porções: T_r e T_e. O conjunto de imagens T_r é utilizado para criar o modelo de decisão $f(\vec{\boldsymbol{x}})$. Já o conjunto T_e é considerado para medir o desempenho de $f(\vec{\boldsymbol{x}})$ por meio de uma métrica de avaliação, por exemplo, a acurácia (veja Seção 2.4.3). Entretanto, em fluxos de dados de imagens, considerando o treinamento online e que as imagens chegam continuamente, não é possível utilizar métodos que consideram a divisão do conjunto de imagens em porções T_r e T_e, para medir o desempenho do classificador. Para tanto, deve-se considerar métodos de avaliação apropriados para fluxos de dados. Alguns exemplos desses métodos estão descritos na Seção 2.4.2.

4.2 Etapas do método de pesquisa

A Figura 8 ilustra as etapas do método de pesquisa considerado neste trabalho de doutorado para a classificação de imagens. A primeira etapa consiste na aquisição das imagens a serem utilizadas para a classificação. Em ambientes de aplicações reais as imagens podem ser obtidas por diferentes meios, tais como: satélites, câmeras e aparelhos médicos.

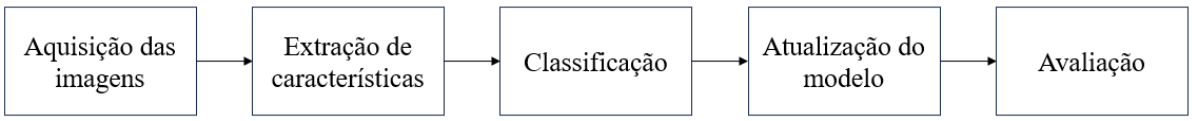

Figura 8 – Etapas do método de pesquisa considerado neste trabalho para a classificação de imagens.

A segunda etapa presente na Figura 8 é relacionada a extração de características das imagens. Para tanto, pode-se utilizar, por exemplo, as técnicas baseadas em BoVW

(*Bag-of-Visual Words*) e CNNs (*Convolutional Neural Networks)*, descritas na Seção 2.1. Após a extração de características das imagens, a terceira etapa do método de pesquisa consiste na classificação das imagens, ou seja, a atribuição de um rótulo para a respectiva imagem analisada.

Conforme descrito na Seção 2.3.1, o classificador utilizado em fluxos de dados de imagens deve possuir características incrementais, isto é, possibilitar um treinamento online do modelo de decisão, com novas imagens que chegam continuamente no fluxo a fim de tratar os fenômenos *concept-drift* e *concept-evolution*. Essa é a quarta etapa ilustrada na Figura 8. Para tanto, pode-se considerar a utilização de técnicas de aprendizado ativo (veja Seção 2.3.2), pois essas técnicas podem ser capazes de identificar instâncias de dados informativas, sendo possível reduzir a quantidade de rótulos solicitados. Por fim, a última etapa presente na Figura 8 consiste na avaliação do classificador. Conforme descrito nas Seções 2.4 e 2.4.2 devem ser considerados métodos de avaliação apropriados para fluxos de dados, que consideram aspectos de cenários de aplicações reais, tais como: ocorrência dos fenômenos *concept-drift* e *concept-evolution*, disponibilidade limitada de rótulos e latência.

4.3 Trabalho desenvolvido

Considerando as etapas do método de pesquisa apresentadas na Seção 4.2 e o objetivo geral desse trabalho de doutorado, descrito no Capítulo 1: *contribuir para a classificação de fluxos de dados de imagens nas etapas de classificação, atualização do modelo e avaliação considerando aspectos inerentes de cenários de aplicações reais*, o trabalho desenvolvido abordou diferentes etapas da classificação de fluxos de dados de imagens, que foram exploradas em capítulos distintos da tese. A Seção 4.3.1 descreve como a a etapa de extração de características foi explorada neste trabalho. A Seção 4.3.2 descreve a respeito da etapa de avaliação. A Seção 4.3.3 detalha como as etapas de classificação e atualização do modelo foram exploradas no trabalho desenvolvido. Por fim, sobre a etapa de aquisição das imagens utilizou-se no decorrer do trabalho, em diferentes experimentos, conjuntos de imagens comumente utilizados na literatura científica da área: *CIFAR10* (AVILA et al., 2013), *CIFAR100* (REBUFFI et al., 2017), *CINIC* (WANG et al., 2019), *COCO* (JAIN et al., 2022), *FASHION-MNIST* (WANG et al., 2019), *Flowers17* (AVILA et al., 2011), *ImageNet* (KÄDING et al., 2017), *LSUN* (WANG et al., 2019), *Pascal VOC 2007* (AVILA et al., 2013), *Scenes15* (LAZEBNIK; SCHMID; PONCE, 2006), *SVHN* (WANG et al., 2019), *Tiny ImageNet* (BENBRAHIM; BEHLOUL, 2021) e *UIUC Sports* (KWITT; VASCONCELOS; RASIWASIA, 2012). A quantidade de imagens e o número de classes de cada conjunto de imagens estão descritos na Tabela 2.

Tabela 2 – Conjuntos de dados de imagens

Conjunto de imagens	Classes	Imagens
CIFAR10	10	60.000
CIFAR100	100	60.000
CINIC	10	100.000
COCO	80	200.000
Fashion-Mnist	10	70.000
Flowers17	17	1.360
ImageNet	1.000	1.281.167
LSUN	10	100.000
Pascal VOC 2007	20	9.963
Scenes15	15	4.485
SVHN	10	100.000
Tiny Imagenet	200	100.000
UIUC Sports	8	1.574

4.3.1 Etapa de extração de características

A etapa de extração de características das imagens foi explorada no Capítulo 5 do trabalho. Esse capítulo apresenta um um estudo experimental com descritores de características para a representação de imagens na classificação de fluxos de dados imagens. O estudo experimental considerou os descritores BoVW e CNN para a descrição das características das imagens. O objetivo desse estudo foi analisar a utilização de diferentes quantidades de imagens para a construção dos descritores de características, além disso, observar a influência da evolução de descritores de características *(feature-evolution)*, considerando que se trata de um cenário dinâmico. Para a realização dos experimentos com CNN foi considerada a arquitetura VGG16, com os vetores de características de 25,088 dimensões produzidos pela última camada de *pooling*, conhecida como *flatten*. Como método de avaliação foi considerado o mesmo método utilizado no trabalho (REBUFFI et al., 2017). O estudo experimental realizado foi descrito no seguinte artigo:

> *M. C. de Lima, A.J.S de Abreu, E. R. Faria and M. C. N. Barioni. "Evaluating the Construction of Feature Descriptors in the Performance of the Image Data Stream Classification". Iberoamerican Congress on Pattern Recognition (CIARP), 2021, pp: 327-339, doi: 10.1007/978-3-030-93420-0_31.*

Apesar dos resultados obtidos nessa análise possibilitarem observar a influência dos descritores de características na classificação de fluxos de dados de imagens, o método de avaliação utilizado em (REBUFFI et al., 2017) não considera aspectos de aplicações

reais, como é o caso da ocorrência de *concept-evolution*. Além disso, é considerada a disponibilidade imediata de todos os rótulos das imagens. Então, observou-se uma questão de pesquisa em aberto, conforme detalhado na Seção 4.3.2.

4.3.2 Etapa de avaliação

Conforme observado por meio da revisão da literatura correlata, é importante destacar que os métodos de avaliação utilizados em classificadores de fluxos de dados de imagens, considerem aspectos de cenários reais. Entretanto, notou-se que os métodos existentes na literatura não são compatíveis com tais cenários. Dessa forma, no Capítulo 6 é apresentado o *framework* EVISClass para a avaliação de classificadores de fluxos de dados de imagens. O método de avaliação utilizado nesse *framework* é apropriado para ambientes em que não se conhece todas as classes de imagens (*concept-evolution*). Além disso, os conceitos podem mudar ao longo do tempo (*concept-drift*) e não se tem o rótulo de todas as imagens ou esses rótulos podem ser informados com atraso (latência). As medidas de avaliação consideradas nos experimentos com o *framework* EVISClass foram: acurácia e *F-measure*. A acurácia permite analisar a quantidade total de instâncias de dados corretamente classificadas. A *F-measure* permite complementar a análise dos resultados verificando o desempenho em contextos com classes desbalanceadas (veja 2.4.3). Essas medidas foram escolhidas pois foram utilizadas em diversos trabalhos de classificação de fluxos de dados de imagens (REBUFFI et al., 2017; CASTRO et al., 2018; WU et al., 2019). Além disso, a acurácia também foi considerada no trabalho que aborda a utilização de *Delayed Label* na classificação em fluxos de dados (GRZENDA; GOMES; BIFET, 2019). É importante ressaltar que as medidas acurácia e *F-measure* também foram consideradas no trabalho descrito em (JAISWAL, 2021), que explorou diferentes medidas de avaliação em classificadores que consideram aprendizado online. Para a complementação da análise do desempenho preditivo dos algoritmos de classificação de fluxos de dados de imagens, foram realizadas análises estatísticas para a comparação de diferentes cenários de experimentos. Para tanto, foi aplicado o *Wilcoxon signed-rank test* (ROSNER; GLYNN; LEE, 2006), um teste estatístico não paramétrico utilizado para comparar duas amostras de resultados e verificar se existe uma diferença significativa entre elas. O desenvolvimento do *framework* EVISClass gerou a seguinte publicação:

> *M. C. de Lima, M. C. N. Barioni, E. R. Faria and H. L. Razente, "EVISClass: a new evaluation method for image data stream classifiers,"International Conference on Machine Learning and Applications (ICMLA), 2020, pp. 399-406, doi: 10.1109/ICMLA51294.2020.00070.*

No Capítulo 6 também foi realizada uma análise semelhante ao Capítulo 4.3.1, mas considerando o *framework* EVISClass como método de avaliação. Além disso, considerou-se

somente o descritor de características CNN nessa análise. Os resultados desse estudo foram publicados no seguinte artigo:

> *M. C. de Lima, Y. S. e Souza, E. R. Faria and M. C. N. Barioni. "A comprehensive analysis of the diverse aspects inherent to image data stream classification". Knowledge and Information Systems 64, 2215–2238 (2022), doi: 10.1007/s10115-022-01717-1.*

4.3.3 Etapas de classificação e atualização do modelo

A etapa de classificação foi abordada por meio da criação de um novo algoritmo para a classificação de fluxos de dados de imagens, denominado HubISC *(Hubness for Image data Stream Classification)*. Esse algoritmo foi apresentado no Capítulo 7 e foi baseado em *hubs* (veja Seção 2.2), para a classificação de fluxos de dados de imagens. O aspecto *hubness* foi considerado por se tratar de uma característica inerente de dados de alta dimensão, como geralmente é o contexto de imagens, mas também pela possibilidade a ser explorada de utilizar *hubs* como uma estratégia de aprendizado ativo e para a sumarização de instâncias de dados. O algoritmo HubISC também apresenta parâmetros que podem influenciar na quantidade de rótulos solicitados, o que pode ser interessante dependendo da disponibilidade de um usuário especialista para fornecimento de rótulos. Os resultados iniciais do algoritmo HubISC foram descritos no seguinte artigo:

> *M. C. de Lima, E. R. Faria and M. C. N. Barioni. "HubISC: um novo algoritmo baseado em hubness para a classificação de fluxo de dados de imagens". Simpósio Brasileiro de Bancos de Dados (SBBD), 2022, pp. 138-150. Búzios. DOI: 10.5753/sbbd.2022.224318.*

4.4 Considerações Finais

Este capítulo descreveu formalmente o problema de classificação de fluxos de dados de imagens. Além disso, foram apresentadas as diferentes etapas envolvidas nessa tarefa. Também foram detalhados os próximos capítulos desta tese, evidenciando como cada etapa da tarefa de classificação de fluxos de dados de imagens foi explorada no trabalho desenvolvido. A partir das diferentes etapas abordadas, é possível perceber que este trabalho apresenta diferentes contribuições para a classificação de fluxos de dados de imagens, que estão detalhadas nos Capítulos 5, 6 e 7.

Capítulo 5

Avaliação de descritores de características para a classificação de fluxos de dados de imagens

A classificação de fluxos de dados de imagens apresenta vários desafios, por exemplo, a evolução de conceitos das classes conhecidas (*concept-drift*) e o surgimento de novas classes (*concept-evolution*). Muitos estudos sobre classificação de fluxos de dados de imagens investigam sobre o classificador, mas não exploram outras questões importantes, por exemplo, sobre a representação das imagens. Com a ocorrência dos fenômenos *concept-drift* e *concept-evolution* pode ser necessário realizar a atualização do descritor de características, além de atualizar o modelo de decisão. Dessa forma, este capítulo apresenta um estudo experimental sobre o impacto das imagens utilizadas para a construção dos descritores de características BoVW e CNN na eficácia dos classificadores de fluxos de dados de imagens. A Seção 5.1 apresenta o estudo experimental sobre descritores de características e a Seção 5.2 apresenta as considerações finais deste capítulo.

5.1 Estudo experimental dos extratores de características

Este estudo experimental tem o objetivo de investigar o uso dos métodos baseados em BoVW e CNN para a representação de fluxos de dados de imagens. Nesses ambientes a construção do vocabulário visual no BoVW e o processo de ajuste de pesos de uma CNN são realizados com base em uma amostra de dados. Essa amostra pode não ser suficientemente representativa para todas as imagens do fluxo, em especial, porque a distribuição de dados muda e novas classes podem surgir com características diferentes das imagens já vistas. Para tanto, pode-se considerar recursos para a evolução dos descritores de características (*feature-evolution*).

A evolução dos descritores de características foi inicialmente abordada nos trabalhos descritos em (REBUFFI et al., 2017; CASTRO et al., 2018). Entretanto, apesar dos resultados promissores obtidos, não foi feita uma análise exaustiva sobre como os parâmetros para a construção do descritor, tais como a quantidade de imagens por classe e o número de classes de imagem, afetam o desempenho do classificador. Além disso, o descritor de características BoVW, considerado um dos estados da arte em imagens, não foi considerado.

Para avaliar a influência dos métodos de extração de características BoVW e CNN na classificação de fluxos de dados de imagens (hipótese de pesquisa **H1**, descrita no Capítulo 1), deseja-se investigar diferentes cenários. Um dos cenários consiste em construir os descritores de características com parte das imagens que serão utilizadas no fluxo de imagens, mas com todas as classes presentes. Em outro cenário de experimentos, deseja-se utilizar um número inferior de classes de imagens para o treinamento dos métodos de extração de características. Com isso, pode-se verificar a capacidade de generalização dos métodos para a descrição de imagens com características diferentes das utilizadas no treinamento inicial, o que pode impactar no resultado do classificador. Para tanto, foi definido o método experimental descrito na Seção 5.1.1.

5.1.1 Método Experimental

O conjunto de imagens X, aqui chamado de conjunto inicial, é um conjunto finito, que possui uma quantidade n_C de diferentes classes de imagens. Cada imagem $i_i \in X$ é descrita por ε, resultando em um vetor de características $\vec{\boldsymbol{x}}_i$. Vale destacar que ao longo do fluxo de dados S novas classes de imagens podem surgir, fazendo com que S possua uma quantidade de classes de imagens $n'_C > n_C$. Além disso, a quantidade d de características das imagens pode não ser suficientemente representativa para a representação das novas imagens que surgem em S.

Com o surgimento de novas classes de imagens e novas instâncias de imagens com características diferentes em cada *timestamp* de S, pode ser necessário a evolução do descritor de características ε para considerar as novas características das imagens. Assim, o objetivo do método experimental descrito aqui é realizar diversas variações da quantidade de classes de imagens e do número de imagens do conjunto inicial (X), para avaliar a influência desses parâmetros na representação do fluxo de dados de imagens S. A partir dessas variações, também pode ser possível observar a influência da descrição de imagens na qualidade do resultado da classificação de imagens. As etapas do método experimental considerado estão ilustradas na Figura 9.

A Figura 9 mostra que o fluxo de dados de imagens (X) é organizado em *batches*. O número de *batches*, a quantidade de classes de imagens e o número de imagens pertencentes a cada *batch* são opções parametrizadas. Essa organização de S utiliza a mesma estratégia descrita em (REBUFFI et al., 2017). Essa estratégia considera a disponibilidade de todos os rótulos das imagens de S. O primeiro *batch* representa o conjunto inicial (X), que é

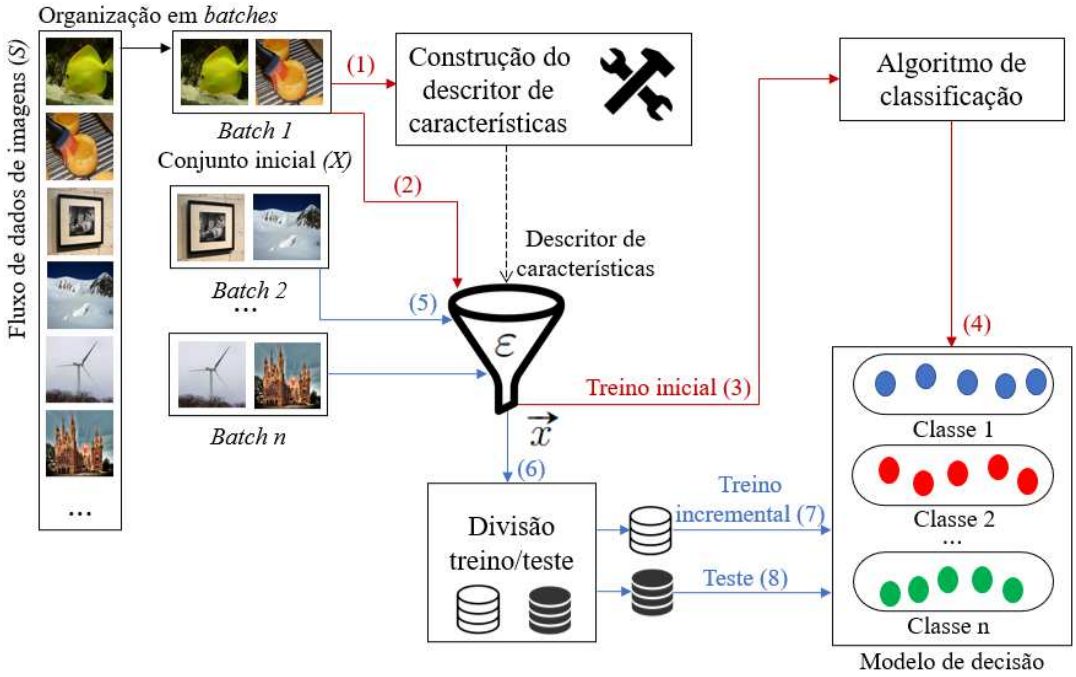

Figura 9 – Etapas do método experimental. Adaptado de (REBUFFI et al., 2017).

utilizado como entrada da Etapa (1) do método experimental. Nessa etapa o descritor de características (ε) é construído. Na Etapa (2) é realizada a extração das características das imagens de X a partir de ε. O treinamento do algoritmo de classificação é realizado após a extração de características na Etapa (3), utilizando como entrada os vetores de características ($\vec{\boldsymbol{x}}$) obtidos na Etapa (2), resultando no modelo de decisão na Etapa (4).

Após o processamento do primeiro *batch*, que é o responsável por construir o descritor de características e também o modelo de decisão, os demais *batches* são considerados. Na Etapa (5), a extração de características das imagens de cada *batch* é realizada considerando ε. Na sequência, os vetores $\vec{\boldsymbol{x}}$ das imagens de cada *batch* são divididos em porções de treino e teste na Etapa (6). Após essa etapa, o treinamento online do modelo de decisão é realizado na Etapa (7). Por fim, o modelo de decisão é testado na Etapa (8). Nota-se que as Etapas de (5) a (8) devem ser realizadas para cada *batch*.

5.1.2 Experimentos

Para realizar os experimentos foram selecionados quatro conjuntos de dados reais, comumente utilizados em trabalhos de classificação de imagens: *Scenes15*, UIUC Sports, *Flowers17* e *Pascal VOC 2007* (veja Seção 4.2).

Os descritores BoVW e CNN foram considerados para a extração de características. A CNN considerada para a realização dos experimentos foi a VGG16 (veja Seção 4.3.1). Para a descrição das imagens considerando BoVW foi realizada uma implementação na linguagem *Java*. Para tanto, o descritor SIFT foi utilizado para a identificação dos pontos-chave (*key points*) das imagens. Considerou-se uma implementação do SIFT disponível

na biblioteca *OpenCV* da linguagem *Python 3.7*. Para o agrupamento dos pontos-chave identificados, o algoritmo *Float k-means* (HARE; SAMANGOOEI; DUPPLAW, 2011) foi utilizado. Para cada conjunto de imagens foram consideradas diferentes quantidades de palavras visuais, baseando-se em trabalhos da literatura científica da área. Para os conjuntos *Flowers17* e *UIUC Sports* foram consideradas 512 palavras visuais (AVILA et al., 2011; KWITT; VASCONCELOS; RASIWASIA, 2012). No conjunto *Scenes15* foram consideradas 800 palavras visuais (LAZEBNIK; SCHMID; PONCE, 2006). Por fim, no conjunto *Pascal VOC 2007* foram consideradas 8.192 palavras visuais (AVILA et al., 2013).

Os algoritmos considerados para a classificação foram o NCM (veja Seção 2.3.3) e o algoritmo iCaRL (REBUFFI et al., 2017) que é uma variação do algoritmo NCM (veja Seção 3.1). O algoritmo iCaRL também possui uma etapa de descrição de características das imagens, sendo que esse método também permite a evolução do descritor de características com novas imagens. A implementação considerada e as configurações utilizadas para o algoritmo iCaRL foram as mesmas publicadas em (REBUFFI et al., 2017). A eficácia dos descritores de características para a classificação foi avaliada considerando duas métricas: acurácia e *F-measure* (veja Seção 2.4.3).

5.1.3 Resultados Experimentais

Considerando o método experimental considerado, descrito na Seção 5.1.1, os experimentos foram realizados considerando 3 diferentes cenários em relação a um caso base. Os cenários de experimentos propostos têm como objetivo analisar diferentes parametrizações, promovendo combinações ainda não exploradas em trabalhos anteriores, para a construção dos descritores de características BoVW e CNN. Essas parametrizações são relacionadas à composição dos conjuntos de imagens utilizados para a construção dos descritores. Além disso, o algoritmo iCaRL foi considerado para permitir analisar os resultados de um método que considera a evolução do descritor de características das imagens. O primeiro cenário de experimentos utiliza diferentes quantidades de imagens, mas todas as classes de imagens são consideradas. O segundo cenário de experimentos analisa a redução do número de classes de imagens. Por fim, o terceiro cenário de experimentos analisa a redução simultânea do número de classes e de imagens. O único cenário explorado em (REBUFFI et al., 2017) foi a redução do número de classes. Portanto, a redução do número de imagens e a limitação simultânea do número de imagens e da quantidade de classes de imagens não foi explorada nos estudos correlatos.

5.1.3.1 Caso base

No caso base de experimentos, os descritores de imagens, BoVW e CNN, consideraram todas as imagens do conjunto de dados para compor o conjunto de imagens inicial (X) para a formação do vocabulário visual e para o treinamento da CNN, respectivamente.

O objetivo é obter uma linha de base para as análises propostas considerando um cenário em que se conhece todas as classes de imagens e se tem a disposição todas as imagens. No algoritmo iCaRL também foram informadas todas as imagens para o descritor de características. Portanto, o recurso de evolução de características do algoritmo iCaRL não foi considerado nesse cenário. Em relação à etapa de classificação do algoritmo iCaRL e do algoritmo de classificação NCM, foram selecionadas, aleatoriamente, as imagens de cada *batch*, sendo 70% das imagens para o treino e 30% para o teste dos classificadores. Cada *batch* possui imagens de 2 classes de imagens. Os resultados dos experimentos com o caso base estão descritos na Tabela 3.

Tabela 3 – Acurácia (A) e *F-Measure* (M) do classificador NCM para BoVW e CNN. (A) e (M) do algoritmo iCaRL. Todas as imagens foram utilizadas para a construção dos descritores.

	NCM				iCaRL	
	BoVW		CNN			
Conjunto Imagens	(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	36,51	37,67	83,08	84,03	82,49	82,61
Scenes15	49,92	48,99	84,76	84,73	83,22	83,15
UIUC Sports	37,81	37,71	90,75	90,52	90,47	90,04
Pascal VOC 2007	27,36	28,3	67,19	65,45	66,74	65,12

Analisando os resultados apresentados na Tabela 3 é possível constatar que o conjunto de imagens *Pascal VOC 2007* é o que se apresentou mais desafiador, tanto para o BoVW, quanto para a CNN. O mesmo comportamento foi observado em (HUANG; WANG; TAO, 2015). Esse conjunto de imagens é o que possui a maior quantidade de imagens e também o maior número de classes. Os resultados do algoritmo iCaRL foram próximos aos resultados do algoritmo NCM com o descritor VGG16. Em ambos os casos os descritores são baseados em CNN. Além disso, o método de evolução de características do iCaRL não foi considerado nesse cenário. Então, nessas circunstâncias, a semelhança dos resultados entre NCM e iCaRL é compatível com os resultados publicados em (REBUFFI et al., 2017). Com a intenção de analisar a influência de diferentes quantidades de imagens utilizadas na construção dos descritores, foram realizados os três cenários de experimentos descritos a seguir.

5.1.3.2 Cenário 1 - Reduzir o número de imagens para a construção dos descritores impacta na capacidade de representação de todo o fluxo de dados de imagens?

Nesse cenário de experimentos o objetivo foi avaliar o impacto da redução do número de imagens do conjunto inicial (X), para a formação do vocabulário visual do BoVW e para o treinamento da CNN. O número de classes de imagens foi preservado,

ou seja, todas as classes foram consideradas. O número de imagens de cada classe foi reduzido para 70%, 50% e 30% em relação ao caso base. As imagens foram selecionadas aleatoriamente, sendo que cada conjunto de imagens inclui os conjuntos inferiores, isto é, o conjunto de 70% inclui as mesmas imagens utilizadas com 50%, que também inclui as imagens de 30%. Conforme descrito na Seção 5.1.1, as imagens utilizadas para o treino do classificador correspondem as mesmas imagens consideradas pelos descritores. Para o algoritmo iCaRL as imagens selecionadas foram utilizadas no primeiro *batch*. Os resultados dos experimentos com esse cenário estão descritos na Tabela 4.

Tabela 4 – Acurácia (A) e *F-Measure* (M) do classificador NCM para BoVW e CNN. (A) e (M) do algoritmo iCaRL. Diferentes quantidades de imagens foram utilizadas para a construção dos descritores.

Conjunto Imagens	Instâncias	NCM				iCaRL	
		BoVW		CNN			
		(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	70%	**35,78**	**36,73**	**80,63**	**82,02**	**82,42**	**82,45**
	50%	**35,78**	36,59	80,14	81,51	82,35	82,31
	30%	35,29	36,24	79,9	81,46	82,2	82,25
Scenes15	70%	**50,22**	**48,96**	**82,98**	82,53	**83,16**	**83,11**
	50%	49,55	48,32	82,17	**82,72**	83,12	82,99
	30%	49,48	48,27	81,65	81,36	83,09	82,81
UIUC Sports	70%	**37,81**	**37,56**	**90,75**	89,89	**90,33**	**90,21**
	50%	37,6	37,52	90,33	**90,04**	90,27	90,15
	30%	37,39	37,42	90,33	89,67	90,27	90,11
Pascal VOC 2007	70%	**27,33**	27,83	**66,72**	**64,62**	**66,73**	**65,78**
	50%	27,09	**28,02**	64,5	63,42	66,69	65,56
	30%	27,23	27,58	64,57	62,14	66,66	65,52

Observando a Tabela 4 é possível constatar que os resultados de acurácia e *F-measure* degradaram para todos os conjuntos de imagens, tanto para o descritor BoVW, quanto para a CNN. No geral, observou-se uma pequena degradação dos resultados de classificação com BoVW, quando comparamos os cenários que consideram 70% e 30% de imagens. A degradação máxima observada foi de 0,69% para a *F-measure* do conjunto de imagens *Scenes15*. Para a CNN foram observadas degradações que chegaram a 2,48% na *F-measure* do conjunto de imagens *Pascal VOC 2007*. Para o algoritmo iCaRL, a maior degradação observada foi de 0,26% para a *F-measure* do conjunto de imagens *Pascal VOC 2007*. Nota-se que a utilização de um método que considera a evolução de características pode amenizar a degradação dos resultados.

Quando comparamos os resultados da Tabela 4 com o caso base, presente na Tabela 3, é possível observar maiores degradações da *F-measure*. Por exemplo, analisando os resultados obtidos para o conjunto *Scenes15* é possível notar uma degradação de até 3,37%

no descritor CNN. Para o BoVW, a maior degradação observada foi de 1,43% no conjunto de imagens *Flowers17*. Esse mesmo conjunto também apresentou a maior degradação no algoritmo iCaRL, com o valor 0,36%. Esses resultados mostram que a redução do número de imagens para a construção dos descritores prejudicou a capacidade de representação de todo o fluxo de dados de imagens.

5.1.3.3 Cenário 2 - Reduzir o número de classes de imagens para a construção dos descritores impacta na capacidade de representação de todo o fluxo de dados de imagens?

O objetivo desse cenário de experimentos foi avaliar o efeito da redução do número de classes de imagens para a formação do vocabulário visual do BoVW e para o treinamento da CNN. Em relação ao caso base, o número de classes de imagens foi reduzido para 70%, 50% e 30%. As classes consideradas foram selecionadas aleatoriamente. Mas os experimentos com 70% de classes incluem também as mesmas classes do caso com 50%, que incluem as classes de 30%. Todas as imagens das classes selecionadas foram consideradas. Com isso, no algoritmo de classificação NCM foram considerados os mesmos conjuntos de treino e teste do caso base. Foram selecionadas, aleatoriamente, 70% das imagens para o treino e 30% para o teste, incluindo as imagens das classes de imagens não utilizadas para a construção dos descritores. Da mesma forma que no cenário anterior, para o algoritmo iCaRL as imagens selecionadas foram utilizadas no primeiro *batch* e as demais imagens foram consideradas para o treinamento online do classificador e também para a evolução do descritor de características. Os resultados dos experimentos com esse cenário estão descritos na Tabela 5.

A partir da análise da Tabela 5 nota-se que existe uma degradação a cada redução do número de classes. Da mesma forma que no cenário anterior, existe uma pequena degradação dos resultados da *F-measure* com o descritor BoVW, sendo constatado no máximo 0,26% de degradação no conjunto de imagens *UIUC Sports* entre os casos 70% e 30% de classes de imagens. Para a CNN, notou-se uma degradação máxima de 0,58% no conjunto de imagens *Pascal VOC 2007*. No algoritmo iCaRL a maior degradação foi observada no conjunto *UIUC Sports*, com o valor de 0,51%.

Comparando os resultados apresentados na Tabela 5 e os resultados do caso base, descrito na Tabela 3, é possível observar maiores degradações da *F-measure*. Para o descritor BoVW, a maior degradação observada foi de 1,98% no conjunto de imagens *Flowers17*. Já para o descritor CNN, a maior degradação foi de 4,27% no conjunto de imagens *Pascal VOC 2007*. No algoritmo iCaRL, a maior degradação foi observada no conjunto *UIUC Sports*, com o valor de 0,41%. No geral, também foi possível observar que a redução do número de classes possui maior influência na qualidade de representação do fluxo de dados de imagens, do que somente a redução do número de imagens, abordada no Cenário 1 de experimentos.

Tabela 5 – Acurácia (A) e *F-Measure* (M) do classificador NCM para BoVW e CNN. (A) e (M) do algoritmo iCaRL. Diferentes quantidades de classes foram utilizadas para a construção dos descritores.

Conjunto Imagens	Classes	NCM				iCaRL	
		BoVW		CNN			
		(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	70%	**35,53**	35,91	**80,63**	**81,29**	**82,27**	**82,31**
	50%	34,8	**36**	80,14	81,28	82,13	82,18
	30%	34,55	35,69	79,65	80,95	81,98	82,01
Scenes15	70%	**49,48**	48,22	**81,13**	**81,01**	**83,12**	**83,09**
	50%	49,33	**48,24**	80,99	81,18	83,07	82,88
	30%	49,4	48,15	80,99	80,76	83,03	82,75
UIUC Sports	70%	**37,39**	**37,32**	**90,12**	**89,68**	**90,21**	**90,07**
	50%	**37,39**	37,26	89,91	89,55	90,15	89,97
	30%	37,18	37,06	89,7	89,41	90,02	89,56
Pascal VOC 2007	70%	**27,23**	**27,57**	**63,83**	**61,75**	**66,67**	**65,51**
	50%	27,13	27,54	62,94	61,5	66,55	65,38
	30%	26,93	27,57	62,82	61,17	66,51	65,25

5.1.3.4 Cenário 3 - Reduzir o número de imagens e a quantidade de classes de imagens para a construção dos descritores impacta na capacidade de representação de todo o fluxo de dados de imagens?

No último cenário de experimentos, o objetivo foi avaliar a junção dos cenários anteriores, isto é, reduzir simultaneamente o número de classes de imagens e a quantidade de imagens. O número de classes utilizadas para a formação do vocabulário visual do BoVW e para o treinamento da CNN foi reduzido para 70%, 50% e 30%. O número de imagens selecionadas também sofreu redução. Foram consideradas as mesmas porcentagens (70%, 50% e 30%). As classes consideradas foram selecionadas com o mesmo critério do Cenário 2. Para a seleção das imagens a mesma estratégia do Cenário 1 foi considerada. Para o algoritmo de classificação NCM foram considerados para o treino os mesmos conjuntos de imagens utilizados para a construção dos descritores. Entretanto, foi necessário realizar o treinamento online com as imagens de classes não conhecidas pelo primeiro conjunto de imagens. Para o treinamento online foram selecionadas as mesmas porcentagens de imagens dos respectivos conjuntos considerados, as demais imagens foram utilizadas no teste do classificador. Para o algoritmo iCarl, as imagens selecionadas foram apresentadas no primeiro *batch* e as demais imagens nos *batches* seguintes para o treinamento online do classificador e para a evolução do descritor de características. Os resultados dos experimentos estão descritos na Tabela 6.

Observando a Tabela 6 foi possível constatar que existe uma degradação gradativa dos resultados a cada redução de classes e também de imagens. Considerando somente

Tabela 6 – Acurácia (A) e *F-Measure* (M) do classificador NCM para BoVW e CNN. (A) e (M) do algoritmo iCaRL. Diferentes quantidades de classes e de imagens foram utilizadas para a construção dos descritores.

Conjunto Imagens	Classes	Instâncias	NCM				iCaRL	
			BoVW		CNN			
			(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	70%	70%	**35,29**	**35,46**	**79,41**	**80,89**	**81,98**	**82,01**
		50%	34,55	35,57	79,65	80,61	81,91	81,94
		30%	34,8	35,49	79,16	80,66	81,69	81,63
	50%	70%	**35,29**	**35,46**	**79,6**	**80,51**	**81,76**	**81,68**
		50%	34,55	35,35	78,18	79,86	81,25	81,47
		30%	34,31	35,37	78,67	79,55	81,25	81,43
	30%	70%	34,06	35,28	78,67	76,13	80,88	81,15
		50%	34,31	34,94	73,52	76,13	80,8	81,07
		30%	34,06	34,92	72,54	73,6	80,73	80,98
Scenes15	70%	70%	**49,11**	**47,76**	**80,47**	80,07	**82,94**	**82,6**
		50%	48,66	47,68	79,95	**80,14**	82,92	80,56
		30%	48,74	47,38	79,51	79,79	82,78	82,19
	50%	70%	48,74	47,37	79,06	79,07	82,83	82,33
		50%	48,59	47,25	77,95	77,97	82,76	82,17
		30%	48,44	47,19	76,77	76,67	82,72	82,08
	30%	70%	48,37	47,1	74,18	74,38	82,63	82,06
		50%	48,15	46,93	70,26	69,56	82,58	81,99
		30%	47,92	46,6	65,9	64,18	82,54	81,94
UIUC Sports	70%	70%	**37,39**	**37,18**	**89,7**	**89,33**	**90,08**	**89,97**
		50%	37,18	36,98	89,28	88,79	90,02	89,88
		30%	37,18	36,91	89,07	88,47	89,89	89,78
	50%	70%	36,97	36,77	88,86	88,25	89,96	89,81
		50%	36,97	36,73	88,02	87,64	89,83	89,61
		30%	36,97	36,7	87,81	87,42	89,77	89,58
	30%	70%	36,97	36,69	86,76	85,96	89,64	89,51
		50%	36,97	36,61	86,13	85,27	89,45	89,33
		30%	36,55	36,28	85,92	85,25	89,39	89,31
Pascal VOC 2007	70%	70%	**27,23**	**27,57**	62,79	**61,15**	**66,49**	**65,18**
		50%	27,13	27,54	**63,12**	60,77	66,47	65,15
		30%	27,13	27,5	62,35	60,61	66,43	65,1
	50%	70%	26,99	27,5	62,52	60,43	66,45	65,13
		50%	26,62	27,5	60,37	59,08	66,34	65,08
		30%	26,56	27,2	60,24	58,96	66,24	65,03
	30%	70%	26,19	27,08	60,64	58,83	65,74	64,99
		50%	26,29	26,9	60,57	57,78	65,64	64,97
		30%	25,92	26,83	59,1	57,14	65,6	64,91

a redução da quantidade de imagens já é possível observar degradações nos resultados. Entretanto, os menores resultados observados foram obtidos para os casos de 30% de classes e 30% de imagens. Comparando a *F-measure* dos casos extremos da Tabela 6, isto é, 70% classes com 70% imagens e 30% classes com 30% imagens, as maiores degradações observadas foram de 1,16% para o BoVW e 15,89% para a CNN. Essas degradações foram observadas no conjunto de imagens *Scenes15*. Para o algoritmo iCaRL, observou-se as menores degradações, com o valor máximo observado de 1,03% no conjunto *Flowers17*.

Em relação ao caso base, descrito na Tabela 3, foram observadas as maiores degradações da *F-measure* em relação a todos os cenários de experimentos. A degradação de 2,75% foi observada para o descritor BoVW no conjunto de imagens *Flowers17*. Para o descritor CNN a maior degradação foi de 20,55% no conjunto de imagens *Scenes15*. No algoritmo iCaRL, a maior degradação observada foi de 1,63% no conjunto *Flowers17*.

A partir da análise realizada, constatou-se que a redução simultânea do número de classes de imagens e da quantidade de imagens provocou uma degradação ainda maior dos resultados se comparado aos cenários de experimentos anteriores. Essa degradação foi observada, em especial, para o descritor CNN. Em todos os conjuntos de imagens, os piores resultados foram observados com as menores quantidades de classes e também de imagens. De forma a complementar a análise dos resultados, foi realizado o teste estatístico descrito na Seção 5.1.3.5.

5.1.3.5 Análise Estatística

A análise estatística realizada comparou os diferentes cenários de experimentos realizados, considerando os valores de acurácia e *F-measure*. Foi aplicado o *Wilcoxon signed-rank test* (veja Seção 4.3.2). A hipótese nula formulada foi: H_0 − os desempenhos preditivos apresentados em dois cenários de experimentos (A e B) não são estatisticamente diferentes, a um nível de confiança de 95%. A hipótese alternativa formulada foi: H_1 − o desempenho preditivo apresentado no cenário de experimentos A é superior ao desempenho preditivo apresentado no cenário de experimentos B.

A primeira análise realizada foi em relação ao Caso Base (Tabela 3) e ao Cenário 1 (Tabela 4), que considera a redução do número de imagens para a construção dos descritores. Os resultados com o descritor BoVW não se mostraram estatisticamente diferentes, isto é, H_0 foi aceita para todos os casos de redução de imagens (70%, 50% e 30%). Esse mesmo comportamento ocorreu para o algoritmo iCaRL. Para o descritor CNN, em todos os resultados foram observadas diferenças estatísticas entre as amostras de resultados. Portanto, H_0 foi rejeitada para o descritor CNN para os casos de 70%, 50% e 30% de imagens, sendo possível concluir que os resultados da Tabela 3 foram superiores com essa configuração.

A segunda análise estatística também considerou o Caso Base (Tabela 3), mas em relação ao Cenário 2 (Tabela 5), que realiza a redução do número de classes de imagens

para a construção dos descritores. Da mesma forma que na análise anterior, H_0 foi aceita para o descritor BoVW, pois não foi constatada diferença significativa entre os resultados obtidos para todos os conjuntos de classes (70%, 50% e 30%). A análise com o descritor CNN mostrou que os resultados possuem significância estatística para todos os conjuntos de classes. Para o algoritmo iCaRL, H_0 foi rejeitada somente para o cenário com 30% de classes. Então, H_0 foi aceita para o BoVW e para os casos de 70% e 50% de classes no algoritmo iCaRL, mas foi rejeitada para o descritor CNN e para o menor conjunto de classes (30%) no algoritmo iCaRL. Sendo possível concluir que os resultados da Tabela 3 foram superiores com as configurações descritas.

Na última análise estatística, os resultados das Tabelas 3 e 6 foram comparados. Diferentemente das análises anteriores, os resultados com o descritor BoVW se mostraram estatisticamente diferentes desde o primeiro caso, isto é, com 70% de classes e 70% de imagens. Para o descritor CNN e para o algoritmo iCaRL também foi observado o mesmo comportamento. Então, H_0 foi rejeitada para todos os experimentos da Tabela 6. Sendo possível concluir que os resultados da Tabela 3 foram superiores em todas as configurações comparadas. Portanto, a partir da análise estatística realizada, conclui-se que a redução simultânea do número de imagens e da quantidade de classes de imagens apresentaram as maiores degradações dos resultados. Nessas condições, até mesmo o iCaRL, que é capaz de atualizar o descritor de características, apresentou degradações estatisticamente diferentes nos resultados. Porém, observa-se que essas degradações são amenizadas em relação aos métodos que não consideram a atualização do descritor de características das imagens.

5.2 Considerações Finais

Este capítulo apresentou um estudo experimental com os descritores de características BoVW e CNN para a representação de imagens na classificação de fluxos de dados imagens. Esse estudo analisou a influência das imagens e classes de imagens consideradas para a construção desses descritores. Por meio da análise dos resultados experimentais obtidos, verificou-se que quanto menor é o número de imagens e de classes de imagens utilizadas na construção dos descritores, maior é a deterioração dos resultados. Além disso, notou-se que a utilização de um algoritmo que considera a evolução de características pode amenizar a degradação da eficácia. Observou-se também que, no geral, com o descritor de características CNN os classificadores apresentaram maiores valores de acurácia e *F-Measure* nos experimentos da Seção 5.1.3. Dessa forma, o descritor de características CNN será considerado nos experimentos realizados nos próximos capítulos deste trabalho.

Para a classificação de imagens, além dos algoritmos NCM e iCaRL, considerados na Seção 3.1, também deseja-se explorar outros algoritmos para comparação, por exemplo, o algoritmo CPE (veja Seção 3.1), que considera a alta dimensionalidade dos dados, característica comum em conjuntos de dados de imagens. Apesar do algoritmo iCaRL considerar

o recurso de *feature-evolution*, esse algoritmo não considera a alta dimensionalidade dos dados. Além disso, recursos de aprendizado ativo não são considerados e o fenômeno *concept-evolution* é ignorado por esse algoritmo. Dessa forma, foi desenvolvido o algoritmo HubISC, descrito no Capítulo 7, que incorpora características ainda não exploradas em outros algoritmos de classificação de fluxos de dados de imagens. O algoritmo HubISC considera o aspecto *hubness* (veja Seção 7.2.1.1) para lidar com dados de alta dimensão. Esse algoritmo também possui recursos de sumarização de instâncias, semelhante ao que ocorre com técnicas baseadas em centróide e medóide. Para tanto, considera-se os *hubs*, instâncias representativas do conjunto de dados, para representarem cada classe de imagens. Além disso, os *hubs* podem contribuir com processos de aprendizado, já que são potenciais instâncias de dados a serem rotuladas por um usuário especialista.

Como método de avaliação, constatou-se na Seção 3.2 que apesar dos trabalhos de classificação de fluxos de dados de imagens considerarem a natureza dinâmica do problema, os algoritmos são avaliados com suposições incompatíveis com cenários de aplicações reais, por exemplo: rótulos fornecidos imediatamente (latência nula), usuário especialista sempre disponível para rotular as imagens e ausência do fenômeno *concept-evolution*. Essas características podem ser observadas no método experimental explorado na Seção 5.1.1, que é baseado no trabalho (REBUFFI et al., 2017). Com o objetivo de suprir essa lacuna, foi desenvolvido o *framework EVISClass*, descrito no Capítulo 6, que é capaz de avaliar os algoritmos de classificação de fluxos de dados de imagens considerando as características de cenários de aplicações reais.

Capítulo 6

Framework de avaliação

Este capítulo apresenta o *framework* EVISClass (*EValuation of Image data Stream Classifiers*) de avaliação de classificadores de fluxos de dados de imagens, que faz parte da investigação da hipótese de pesquisa **H2**, descrita no Capítulo 1. Trabalhos recentes da literatura de classificação de imagens avaliam os classificadores em cenários bem controlados supondo que todas as classes do problema são previamente conhecidas e que os rótulos estão sempre disponíveis para atualizar o modelo, sem qualquer atraso. Dessa forma, esses métodos de avaliação não consideram a presença de latência e a ocorrência dos fenômenos *concept-evolution* e *concept-drift*. Então, podem ser métodos inadequados para cenários de aplicações reais, nos quais diferentes classes de imagens chegam continuamente e não se dispõe de um especialista para rotular todas as imagens. Os detalhes do *framework* EVISClass são descritos na Seção 6.1. A Seção 6.2.1 descreve a avaliação experimental realizada com o *framework* EVISClass. Este capítulo também estendeu o estudo realizado no Capítulo 5 sobre descritores de características. Essa análise experimental foi realizada na Seção 6.2.2. Para tanto, considerou-se o refinamento do descritor de características CNN, utilizando o *framework* EVISClass nos experimentos. Por fim, a Seção 6.3 apresenta as considerações finais deste capítulo.

6.1 EVISClass

O *framework* EVISClass tem o objetivo de fornecer meios para permitir a análise de diversos aspectos inerentes à aplicação de tarefas de classificação de fluxos de dados de imagens. Para tanto, o *framework* EVISClass possui dois componentes principais: um simulador de fluxos de dados de imagens; e um avaliador de classificadores de fluxos de dados de imagens. O Algoritmo 14 apresenta uma visão geral do *framework*.

O simulador de fluxos de dados de imagens permite gerar um fluxo de imagens a partir de um conjunto de imagens considerando diferentes parâmetros. Dado um conjunto de imagens D, no qual cada imagem i_i pertencente a esse conjunto é representada por

Algoritmo 14: Framework EVISClass

Entrada: D Conjunto de imagens, q Quantidade de imagens *batch* inicial, c Quantidade de classes *batch* inicial, *classOption* Forma de surgimento de novas classes, *instanceOption* Forma de surgimento das instâncias, *labelOption* Forma considerada para informação dos rótulos, l Latência, f Modelo de decisão, m Medida de avaliação

Saída: m do modelo de decisão f

```
1  início
2      X ← criação do conjunto rotulado de imagens inicial com c classes e q
       imagens a partir de D considerando instanceOption;
3      S ← criação do fluxo de imagens a partir de D considerando classOption e
       instanceOption;
4      f ← treinamento a partir de X;
5      Z ← ∅;
6      repita
7          recebe x⃗_t por meio de S;
8          y_t ← f(x⃗_t);
9          se o rótulo verdadeiro de x⃗_t deve ser informado considerando
           labelOption então
10             Z ← Z ∪ x⃗_t;
11         se (timestampAtual - timestamp(Z[0])) ≥ l então
12             I ← Z[0];
13             ŷ ← rótulo verdadeiro de I;
14             atualiza f considerando ŷ;
15             remove I de Z;
16             avalia o classificador com base em ŷ e y de I considerando m;
17     até o fim de S ;
18     return m;
19 fim
```

um vetor de características $\vec{\boldsymbol{x}}_i$ e possui uma classe y, o simulador gera dois conjuntos de dados, X e $S \in D$, sendo $S = D - X$.

O conjunto de dados X corresponde a um subconjunto de imagens selecionadas do conjunto de dados D para serem utilizadas para o treinamento inicial do modelo de decisão, aqui também chamado de classificador. Para realizar essa seleção é necessário informar a quantidade de imagens q, sendo $q \leq |D|$, a serem selecionadas para compor X e a quantidade de classes c, sendo $c \leq |n_c|$, de imagens que devem estar presentes em X. Considerando esses parâmetros as imagens são selecionadas a partir de D de acordo com a maneira selecionada (parâmetro *instanceOption* do Algoritmo 14).

O conjunto de dados S contém as demais imagens pertencentes a D que não foram selecionadas para compor X. Para simular um fluxo de imagens, a ordenação das imagens em S pode ser definida considerando diferentes estratégias tanto para o surgimento de novas classes (parâmetro *classOption* do Algoritmo 14), quanto para o surgimento de instâncias de imagens de cada uma dessas classes (parâmetro *instanceOption* do Algoritmo

14).

O *framework* EVISClass disponibiliza duas opções para o surgimento de novas classes: surgimento aleatório das classes no fluxo de imagens S sem restrições, isto é, em qualquer *timestamp* podem chegar exemplos de qualquer classe; e surgimento aleatório de novas classes no fluxo de imagens S considerando uma janela fixa de tamanho n, isto é, a cada n imagens surge uma nova classe no fluxo de imagens, sendo $n = |S|/m$, tal que m corresponde ao número de classes de D, mas que não estão em X. Em ambas as opções o sorteio aleatório é feito seguindo uma distribuição uniforme.

Para o surgimento de instâncias de imagens também existem duas opções: surgimento aleatório; e surgimento considerando uma simulação de mudança de conceito. Para essa última opção, a estratégia desenvolvida seleciona inicialmente as instâncias de imagens de uma dada classe que sejam mais similares ao centróide da classe. Dessa forma, entende-se que o conceito de cada classe muda ao longo do fluxo de imagens, pois as últimas imagens a serem adicionadas ao fluxo de imagens S serão menos similares ao centróide da classe do que as primeiras. Essa estratégia foi inspirada no trabalho descrito em (PUNN; AGARWAL, 2018) e está ilustrada na Figura 10.

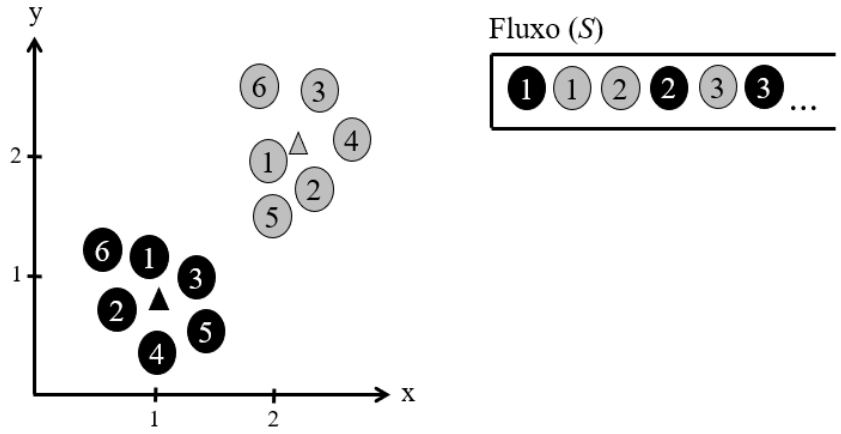

Figura 10 – Estratégia de mudança de conceito considerada pelo *framework*.

A Figura 10 ilustra um conjunto de dados $2D$ que possui duas classes, os centróides de cada classe estão destacados em triângulos. Então, supondo um conjunto de imagens, as primeiras imagens a serem adicionadas ao fluxo são próximas ao seu respectivo centróide. As últimas imagens a serem adicionadas correspondem às bordas mais extremas de cada classe. Esse procedimento de formação do fluxo de imagens também está ilustrado do lado direito da Figura 10. Dessa forma, entende-se que o conceito de cada classe muda ao longo do fluxo de imagens, pois as últimas imagens adicionadas são distantes das primeiras, em termos de similaridade. A Figura 11 ilustra um exemplo de organização de conjunto de dados utilizando o simulador de fluxos de dados de imagens do *framework* EVISClass.

Na Figura 11, verticalmente, cada cor representa uma classe de imagens de um conjunto de imagens com 8 classes. Na horizontal é a ordem de surgimento de cada

instância de dados. As primeiras instâncias de dados, representadas em preto e vermelho, representam o conjunto de dados X. Já as demais instâncias de dados pertencem ao fluxo de dados S. Caso seja considerado o recurso para simular *concept-drift*, as últimas imagens do fluxo são as mais distantes em relação ao centróide das classes. Com essa ilustração da organização do conjunto de dados, é possível perceber algumas características de fluxos de dados, por exemplo, surgimento mais tardio de algumas classes (classe marrom) e desaparecimento prematuro de outras classes (classe azul). Além disso, com esse recurso é possível contrastar o surgimento de novas classes com o desempenho do classificador durante janelas de dados específicas, observando a influência dos fenômenos *concept-evolution* e *concept-drift*.

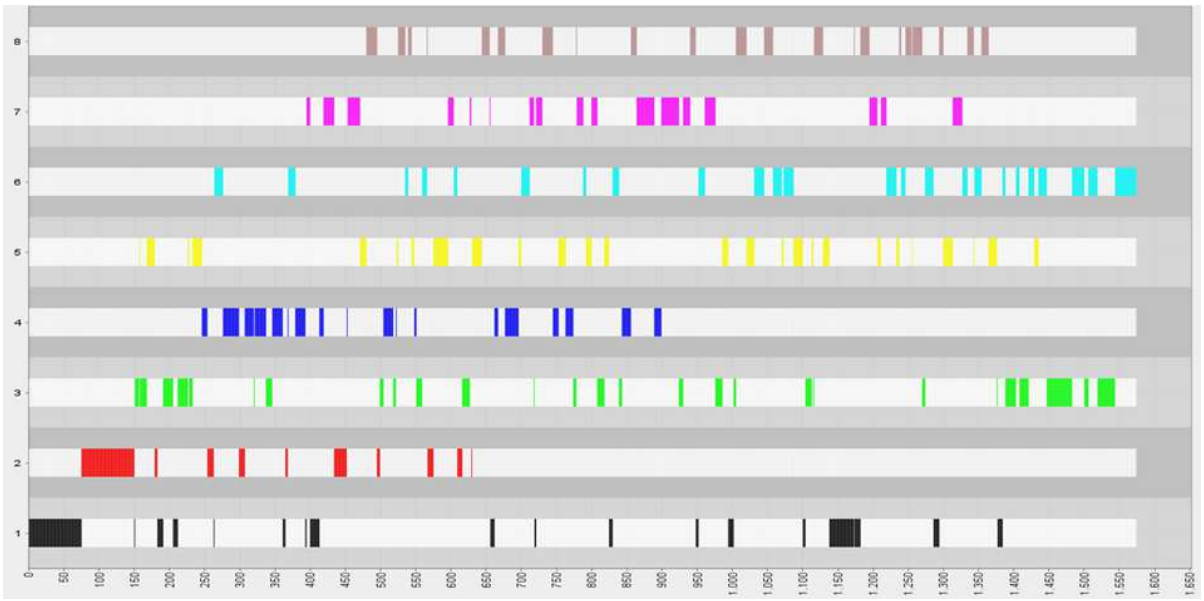

Figura 11 – Exemplo de organização de conjunto de dados com o simulador de fluxos de dados de imagens do *framework* EVISClass.

O avaliador de classificadores de fluxos de dados de imagens do *framework* EVISClass foi desenvolvido considerando as abordagens *Interleaved Test-Then-Train* e *Delayed Label* descritas na Seção 2.4.2. Para realizar a avaliação é necessário informar o modelo de classificação considerado (f). O classificador a ser utilizado pode ser qualquer algoritmo de classificação com características incrementais, por exemplo, o algoritmo NCM descrito na Seção 2.3.3. Para o treinamento do modelo f, utiliza-se o conjunto de imagens X, conforme descrito na linha 4 do Algoritmo 14. Após o treinamento de f, considera-se o fluxo de imagens S para o teste do modelo de decisão. Para cada imagem $\vec{\boldsymbol{x}}_t$ pertencente a S, atribui-se um rótulo y_t a partir de f, conforme descrito na linha 8 do Algoritmo 14.

Para a atualização do modelo de decisão, considera-se uma abordagem semelhante à abordagem *Delayed Label*. Contudo, com o objetivo de diminuir a intervenção do usuário especialista, o *framework* EVISClass disponibiliza estratégias que não requerem a informação de rótulo para todas as imagens, como pode ser observado nas linhas 9 e 10 do Algoritmo 14.

Dentre as diferentes estratégias oferecidas pelo EVISClass para selecionar quais imagens devem ter o rótulo informado estão: todas as imagens devem ter o rótulo informado; definição aleatória se cada imagem deve ter seu rótulo informado ou não; seleção de imagens classificadas com incerteza para terem seus rótulos informados (veja Seção 2.3.2). Independente da estratégia escolhida para a seleção de imagens para serem rotuladas, após a seleção dessas imagens, EVISClass considera a latência l informada como parâmetro de entrada para informar o rótulo $\hat{y}$ de cada imagem (veja linha 11 do Algoritmo 14). Na linha 14 do Algoritmo 14 o modelo de decisão f é atualizado de maneira online considerando $\hat{y}$. Por fim, na linha 16 do Algoritmo 14 o resultado é avaliado com base em uma medida de avaliação m informada como parâmetro do *framework*.

6.2 Experimentos

Os experimentos descritos na Seção 6.2.1 têm o objetivo de aplicar o *framework* EVISClass em cenários similares aos de aplicações reais, nos quais existem: *concept-evolution*, *concept-drift* e a ausência de rótulos para as imagens. Essa análise foi complementada nos experimentos descritos na Seção 6.2.2. Esses experimentos estenderam o estudo realizado no Capítulo 5, utilizando o *framework* EVISClass na análise sobre a construção do descritor de características CNN.

6.2.1 Análise experimental do *framework* EVISClass

Para a realização dos experimentos foi utilizado um subconjunto de imagens do conjunto CIFAR100 (veja Tabela 2). Nos experimentos realizados nessa tese foram consideradas as 20 primeiras classes do conjunto de imagens, totalizando 10.000 imagens.

O *framework* EVISClass pode ser usado para a avaliação de diferentes algoritmos de classificação. Para os experimentos descritos aqui, foi usado o algoritmo de classificação NCM, descrito na Seção 2.3.3. Para a extração de características das imagens também foi considerado o mesmo extrator utilizado no trabalho descrito em (REBUFFI et al., 2017), que considera CNNs. Os parâmetros de configuração do algoritmo de extração de características foram os mesmos utilizados no trabalho correlato (REBUFFI et al., 2017), o que permitiu obter um vetor numérico de 64 dimensões para cada imagem. A medida de avaliação considerada nos experimentos foi a acurácia (veja Seção 4.2).

Os experimentos realizados foram analisados considerando três cenários diferentes em relação aos resultados apresentados por um caso base da literatura correlata (REBUFFI et al., 2017). O primeiro cenário de experimentos empregou o método de avaliação *Interleaved Test-Then-Train*, considerando o *framework* EVISClass com latência nula (igual a 0). O segundo cenário de experimentos utilizou diferentes valores de latência com o intuito de analisar a influência da latência nos resultados do algoritmo de classificação sendo considerado. O terceiro cenário de experimentos teve o objetivo de investigar a

influência da quantidade de rótulos informados nos resultados do algoritmo de classificação sendo avaliado.

6.2.1.1 Caso base

O método de avaliação empregado em (REBUFFI et al., 2017) e descrito na Seção 3.2 considera a divisão das imagens em *batches*. Nesse método as imagens sempre são apresentadas primeiramente para o treino e depois para o teste, não ocorrendo o surgimento de novas classes diretamente na fase de teste. A ilustração desse método de avaliação está presente na Figura 12(A), já o funcionamento do *framework* EVISClass é apresentado na Figura 12(B).

A partir da comparação das Figuras 12(A) e 12(B), nota-se que a abordagem de (REBUFFI et al., 2017) utiliza um mecanismo baseado em *batches*. Os *batches* são compostos por diferentes classes e estão ilustrados por cores distintas na Figura 12(A). Observa-se também, que na abordagem de (REBUFFI et al., 2017) todas as classes de imagens utilizadas no conjunto de testes, já são conhecidas pelo modelo de decisão, o que não permite a ocorrência de *concept-evolution*.

O *framework* EVISClass, ilustrado na Figura 12(B), não considera a divisão do conjunto de imagens em *batches*. As classes de imagens estão ilustradas por cores distintas na Figura 12(B) e podem aparecer aleatoriamente no fluxo de imagens, o que possibilita a ocorrência de *concept-evolution*, pois o treinamento inicial do modelo de decisão pode ter sido realizado com uma pequena quantidade de classes de imagens (ex: classes identificadas com cinza e azul na Figura 12(B)), sendo necessário realizar um treinamento online do modelo de decisão para o aprendizado de novas classes. É importante destacar que no *framework* EVISClass, imagens de classes já conhecidas pelo modelo de decisão podem ser apresentadas novamente no fluxo de imagens, inclusive com a ocorrência de *concept-drift*.

Nos experimentos realizados aqui, que considera método de avaliação empregado em (REBUFFI et al., 2017) (Figura 12(A)), foram utilizados diferentes tamanhos de *batches*, com o objetivo de verificar a influência do tamanho dos mesmos nos resultados. Para cada quantidade de *batches* especificada foi considerada uma quantidade de classes do *batch* inicial diferente. Para todos os cenários avaliados foi considerado o surgimento de 2 classes em cada *batch* incremental. Os resultados obtidos para esse conjunto de experimentos são apresentados na Tabela 7. A acurácia apresentada na Tabela 7 corresponde a média de 10 execuções do algoritmo de classificação. A acurácia de cada execução equivale a média das acurácias apresentadas em cada *batch*, isto é, a cada *batch* apresentado ao classificador, mede-se a acurácia do procedimento de teste do respectivo *batch*.

Analisando a Tabela 7 é possível notar que quanto menor o número de *batches*, maior a degradação dos resultados. O melhor resultado, em termos de acurácia, foi obtido com a configuração de 10 *batches* considerando 2 classes no *batch* inicial, que apresentou 73,5% de acurácia. Esse resultado é compatível com os resultados obtidos pelo trabalho

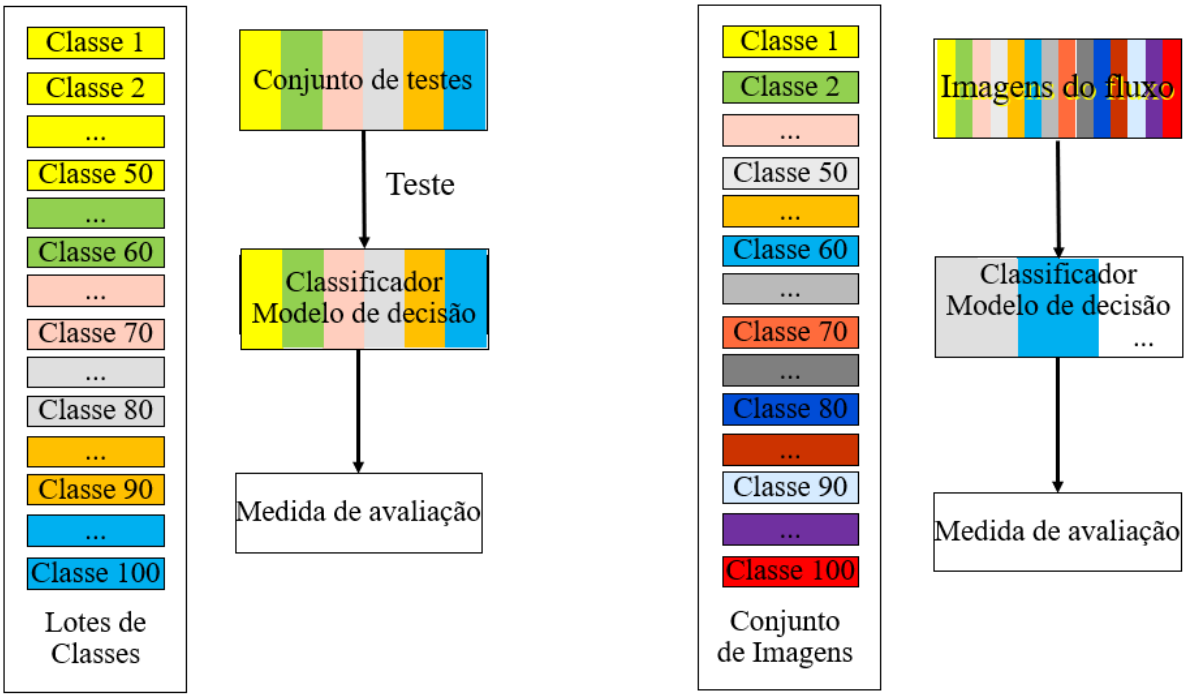

Figura 12 – (A) Ilustração do método de avaliação utilizado em (REBUFFI et al., 2017) em um conjunto de imagens com 100 classes. (B) Ilustração do método de avaliação do *framework* EVISClass.

original descrito em (REBUFFI et al., 2017). Esse fato ocorre pois com um menor número de *batches* mais classes estarão em cada *batch*, o que pode dificultar a classificação e obter um menor valor na acurácia. Dessa forma, nota-se que quanto maior o número de *batches*, maior pode ser a acurácia, já que nos *batches* iniciais, com poucas classes, a tendência é que uma acurácia maior seja obtida.

Tabela 7 – Acurácia do classificador NCM usando a avaliação proposta por (REBUFFI et al., 2017).

Total *batches*	Classes *batch* inicial	(A)
10	2	73,5 ± 0,133
9	4	71,6 ± 0,125
8	6	70,6 ± 0,122
7	8	69,4 ± 0,118
6	10	68,6 ± 0,110
5	12	68,2 ± 0,110
4	14	67,7 ± 0,099
3	16	67,0 ± 0,094
2	18	67,0 ± 0,085
1	20	66,9 ± 0,007

O método de avaliação utilizado pelo trabalho descrito em (REBUFFI et al., 2017) e replicado aqui neste cenário de experimentos não considera o surgimento de novas classes diretamente na fase de teste do classificador. Além disso, ele considera a disponibilidade

de todos os rótulos, latência nula e não trata *concept-drift*. Com o objetivo de analisar o comportamento de um classificador de fluxos de dados de imagens, que é estado da arte, considerando um cenário mais real, o *framework* EVISClass foi utilizado para analisar o desempenho do algoritmo NCM considerando os três cenários de experimentos descritos a seguir.

6.2.1.2 Cenário 1 - Qual é a influência do tamanho do conjunto inicial de imagens e da forma de surgimento de novas classes nos resultados?

Neste cenário de experimentos o objetivo foi simular um fluxo de imagens a partir de um conjunto estático de imagens com a abordagem *Interleaved Test-Then-Train*. Neste cenário as imagens são utilizadas, primeiramente, para o teste do classificador e na sequência são disponibilizadas para o treinamento online do modelo de decisão. Dentre as estratégias oferecidas pelo EVISClass para selecionar quais imagens devem ter o rótulo informado foi utilizada a opção em que todas as imagens devem ter o rótulo informado (parâmetro *labelOption* do Algoritmo 14) com latência nula (parâmetro l do Algoritmo 14).

Conforme descrito na Seção 6.1, um dos módulos principais do *framework* (linhas 2 e 3 do Algoritmo 14) é responsável por reorganizar um conjunto estático de imagens D, para simular um fluxo de dados. Com o objetivo de verificar a influência do tamanho do conjunto inicial de imagens no desempenho do classificador, diferentes valores foram utilizados no *framework* para os parâmetros q (quantidade de instâncias do *batch* inicial) e c (quantidade de classes de imagens do *batch* inicial). Para o parâmetro c foram considerados os valores 2, 4, 6, 8 e 10 (10%, 20%, 30%, 40% e 50% em relação ao conjunto CIFAR com 20 classes). Para o parâmetro q foram utilizadas 125, 250, 375 e 500 imagens para cada classe (25%, 50%, 75% e 100% das instâncias de cada classe, respectivamente).

Além dos parâmetros q e c, variou-se também o parâmetro *classOption* do Algoritmo 14, que corresponde a forma de surgimento das novas classes no fluxo de imagens. Foram consideradas duas abordagens: o surgimento aleatório de novas classes sem restrições; e o surgimento aleatório considerando uma janela fixa de tamanho n. Para o surgimento das instâncias (parâmetro *instanceOption* do Algoritmo 14) utilizou-se a abordagem que simula a mudança de conceitos de cada classe. Os resultados deste cenário de experimentos estão apresentados na Tabela 8.

Analisando o conjunto de resultados obtidos para cada quantidade distinta de classes no *batch* inicial considerando a abordagem com uma janela fixa é possível observar que, de maneira geral, a acurácia aumenta a medida que a quantidade de instâncias utilizadas no *batch* inicial aumenta. Por exemplo, comparando os resultados obtidos para $c = 10$ a acurácia aumentou 7,06% quando se compara $q = 1.250$ e $q = 5.000$. Por outro lado, a medida que a quantidade de classes no *batch* inicial aumenta a acurácia do classificador diminui. Por exemplo, comparando os resultados obtidos quando se considera

Tabela 8 – Acurácia (A) do classificador NCM no *framework* proposto com latência nula.

Classes *batch* inicial (c)	Instâncias *batch* inicial (q)	(A) abordagem aleatória com janela fixa	(A) abordagem aleatória sem restrições
2	250 (25%)	73,1	67,7
	500 (50%)	72,7	67,6
	750 (75%)	72,7	67,3
	1.000 (100%)	**73,3**	68,1
4	500 (25%)	72,0	67,7
	1.000 (50%)	71,0	65,5
	1.500 (75%)	70,0	65,4
	2.000 (100%)	71,6	67,7
6	750 (25%)	70,3	66,3
	1.500 (50%)	68,6	63,6
	2.250 (75%)	68,9	64,0
	3.000 (100%)	72,4	68,6
8	1.000 (25%)	68,8	65,1
	2.000 (50%)	66,4	61,6
	3.000 (75%)	66,7	62,1
	4.000 (100%)	71,8	68,5
10	1.250 (25%)	66,6	68,5
	2.500 (50%)	63,4	59,3
	3.750 (75%)	64,2	60,2
	5.000 (100%)	71,3	68,1

100% das instâncias para $c = 2$ e $c = 10$ a acurácia diminuiu 2,8%.

Além disso, está claro que quando as classes surgem de maneira aleatória, sem restrições, a acurácia do classificador é inferior à acurácia obtida quando as classes surgem com uma janela fixa. Esse comportamento era esperado, pois com a abordagem aleatória, sem restrições, várias novas classes podem surgir a qualquer momento. Já na abordagem com uma janela fixa as novas classes surgem em um intervalo predeterminado de instâncias (veja Seção 6.1).

6.2.1.3 Cenário 2 - Qual é a influência da latência nos resultados?

Neste cenário de experimentos o objetivo foi avaliar o impacto do atraso na informação dos rótulos das imagens na acurácia do classificador. Diferentes valores de latência foram considerados na realização dos experimentos. Os demais parâmetros foram fixados em: um *batch* inicial de 500 imagens (5% das instâncias do conjunto de dados) divididas em 4 classes (20% das classes do conjunto de dados) com a abordagem aleatória sem restrições para o surgimento de novas classes. Os resultados dos experimentos estão descritos na segunda coluna da Tabela 9.

Analisando essa tabela fica evidente que quanto maior o valor da latência maior é a degradação da acurácia do classificador. O primeiro resultado deste cenário de experimentos refere-se a latência nula. Portanto, a acurácia de 67,7% é a mesma apresentada na Tabela

8. A partir dos demais resultados deste cenário, nota-se que a latência possui influência nos resultados da acurácia, quanto maior o valor da latência, mais os resultados degradaram. Além disso, é possível observar também que até a latência 500, a acurácia se manteve acima de 60%.

Tabela 9 – Acurácia do classificador NCM no *framework* EVISClass para os cenários de avaliação 2 e 3.

Latência	Cenário 2	Cenário 3					
		Aleatória		Incerteza		Incerteza com Random.	
	(A)	Rótulos informados	(A)	Rótulos informados	(A)	Rótulos informados	(A)
0	**67,7**	5.243 ± 31	**67,0 ± 0,004**	**729**	63,6	800 ± 3	65,3 ± 0,0008
10	67,6	5.220 ± 35	66,7 ± 0,003	811	64,7	908 ± 11	65,1 ± 0,002
100	66,7	5.149 ± 44	65,0 ± 0,004	1.240	64,2	1.532 ± 12	65,0 ± 0,002
500	62,4	4.752 ± 50	57,5 ± 0,004	2.085	59,6	2.788 ± 14	60,0 ± 0,001
1.000	57,8	4.240 ± 55	48,7 ± 0,004	2.551	55,0	3.612 ± 12	56,8 ± 0,001
2.000	49,2	3.263 ± 22	34,1 ± 0,002	2.897	45,4	4.366 ± 15	48,8 ± 0,0003
3.000	40,9	2.282 ± 50	24,0 ± 0,003	2.861	37,4	4.517 ± 19	40,9 ± 0,0001
4.000	34,1	1.248 ± 41	18,1 ± 0,002	2.769	31,5	4.334 ± 15	34,2 ± 0,0001

6.2.1.4 Cenário 3 - Qual a influência da quantidade de rótulos nos resultados?

O objetivo desse cenário de experimentos foi investigar o impacto da quantidade de rótulos na acurácia do classificador sendo avaliado. Para tanto, foram consideradas três estratégias para selecionar quais imagens devem ter o rótulo informado (parâmetro *labelOption* do Algoritmo 14): definir aleatoriamente as imagens que terão os rótulos informados; selecionar imagens classificadas com baixa confiança, considerando um limiar definido automaticamente pelo algoritmo no treinamento do *batch* inicial; e selecionar imagens rotuladas com baixa confiança, mas com alterações aleatórias do limiar.

A primeira estratégia funciona da seguinte maneira: após o treinamento do classificador com o *batch* inicial, para cada imagem que chega no fluxo, o *framework* EVISClass define aleatoriamente se ela terá ou não seu rótulo informado. Contudo, o rótulo da imagem não é informado imediatamente. As mesmas latências para a informação de rótulo definidas no cenário de experimentos anterior foram consideradas. Os resultados obtidos com a utilização desta estratégia estão apresentados nas colunas 3 e 4 da Tabela 9.

Analisando esses resultados é possível observar a partir de latência 500 fica evidente a degradação mais acentuada da acurácia do classificador. Esses resultados podem ser explicados pelo fato que com valores mais altos de latência, o fluxo de imagens pode se encerrar sem que algumas imagens do *buffer* tenham os seus rótulos informados. É importante ressaltar que o número de rótulos fornecidos e as acurácias obtidas neste cenário de experimentos equivalem a média de 10 execuções do algoritmo.

A segunda estratégia reduz o número de rótulos fornecidos obtendo somente o rótulo de imagens classificadas com incerteza (baixa confiança). Para tanto, considerando que se trata do classificador NCM, no treinamento com o *batch* inicial de imagens, identifica-se a maior distância existente entre uma imagem e o seu respectivo centróide da classe de imagem. Então, essa maior distância é utilizada como um limiar l: caso uma imagem que chega no fluxo possua uma distância maior que l, em relação a todos os centróides, esta imagem deve ter o seu rótulo informado. O limiar também pode ser alterado com a atualização online do modelo de decisão, pois as distâncias das imagens que tiveram seus rótulos fornecidos também são consideradas. Os resultados obtidos com a utilização desta estratégia são mostrados nas colunas 5 e 6 da Tabela 9.

Analisando esses resultados fica evidente que o número de rótulos fornecidos é inferior ao da estratégia com rótulos aleatórios até a latência 2.000. Além disso, à medida que a latência aumenta, o número de rótulos fornecidos também aumenta. Esses resultados são devidos ao fato de que ocorreram menos atualizações online do modelo de decisão, por conta do maior atraso na informação dos rótulos, gerando menos alterações do limiar de distância. Em termos de acurácia, os valores apresentados neste cenário, a partir da latência 500, são superiores em relação aos resultados da abordagem com rótulos aleatórios. Esses resultados mostram que mesmo com uma quantidade inferior de rótulos, a utilização de uma estratégia mais sofisticada para a seleção das imagens que devem ter o rótulo informado, permite a obtenção de melhores resultados em termos de acurácia. Isso indica que é mais importante escolher rótulos adequados para a atualização do modelo de decisão do que usar todos, o que contribui para evitar o *overfitting* do mesmo.

A terceira estratégia para o fornecimento de rótulos permite alterar o limiar, utilizado para determinar a confiança na classificação de uma imagem, de maneira aleatória. Para tanto, uma adaptação da estratégia denominada *Incerteza com Randomização*, definida em (ŽLIOBAITė et al., 2014), foi considerada. Nessa abordagem, divide-se o valor do limiar atual por um valor definido aleatoriamente no intervalo $[1, 2]$. Essa abordagem pode facilitar quando chegam no fluxo imagens com mudança de conceitos das classes (*concept-drift*), pois como o valor do limiar diminui, mais imagens devem ter o rótulo fornecido. Os resultados desta estratégia são mostrados nas colunas 7 e 8 da Tabela 9. Analisando os resultados obtidos é possível notar que o número de imagens com os rótulos fornecidos aumentou em relação a abordagem baseada em incerteza, refletindo nas acurácias obtidas. A acurácia dos resultados também aumentou em até 3,5%.

Analisando todos os cenários investigados, conclui-se que quanto maior a realidade considerada, maior é a degradação da acurácia do classificador avaliado. Quanto maior a latência, menor foram os valores de acurácia. Além disso, ficou evidente que mesmo com uma menor quantidade de rótulos informada, as acurácias obtidas com o uso da abordagem baseada em incerteza são próximas ou superiores as acurácias obtidas com as demais abordagens. Portanto, os dois fatores que mais influenciaram nos resultados foram

a latência empregada e a estratégia de fornecimento de rótulos adotada.

6.2.2 Análise experimental do refinamento do descritor de características CNN considerando o *framework* EVISClass

Os experimentos realizados na Seção 6.2.1 não investigaram questões sobre o descritor de características das imagens. Na Seção 5.1 foi realizada uma análise experimental considerando os descritores BoVW e CNN. Contudo, essa análise experimental não utilizou o *framework* EVISClass para a avaliação dos classificadores de fluxos de dados de imagens. De forma a complementar a análise realizada no Capítulo 5 e estender a investigação da hipótese de pesquisa **H1** (veja Capítulo 1), foi realizada uma análise experimental, com o descritor de características CNN, observando os impactos da quantidade de imagens utilizada para o refinamento do descritor de características para a eficácia dos classificadores. Além disso, foi considerado um algoritmo que implementa *feature-evolution* para a representação das imagens.

Para a realização da análise descrita aqui foi utilizado um método experimental que considera as características do *framework* EVISClass, detalhadas na Seção 6.1, tais como: *I)* surgimento de novas classes (*concept-evolution*), isto é, na etapa de testes surgem novas classes diferentes das disponíveis na fase de treinamento; *II)* evolução dos conceitos de classes já existentes (*concept-drift*); *III)* atualização do modelo de decisão considerando diferentes atrasos (latência) na entrega dos rótulos para atualização do modelo; *IV)* escolha de um subconjunto de instâncias a serem rotuladas para atualização do modelo por meio do uso de técnicas de aprendizado ativo; *V)* refinamento de descritores de características utilizando diferentes quantidades de imagens e de classes de imagens. Uma visão geral desse método experimental está ilustrada na Figura 13.

A Figura 13 mostra que as Etapas (1) e (2) do método experimental são referentes ao refinamento do descritor e a extração de características das imagens. As tarefas realizadas a partir da Etapa (3) são referentes a classificação, obtenção de rótulos e atualização do modelo de decisão. A partir da Figura 13 é possível perceber que um conjunto de imagens X, com c classes e q imagens, é utilizado como entrada da Etapa (1). Nessa etapa o descritor de características ε passa por um processo de refinamento a partir de um conjunto de dados inicial. No caso do descritor de características CNN utilizado nesse experimento, essa etapa se baseia no refinamento da CNN, previamente construída, a partir de outro conjunto de imagens. Geralmente, utiliza-se uma CNN construída em conjuntos de dados com várias classes de imagens, como é o caso do conjunto de imagens *ImageNet*, que possui mais de 1.000 classes. Essa etapa de refinamento também é comumente conhecida como *fine-tuning* (KÄDING et al., 2017).

Após o refinamento do descritor de características, pode-se utilizá-lo para a extração de características das imagens. Então, na Etapa (2) do método experimental é realizada

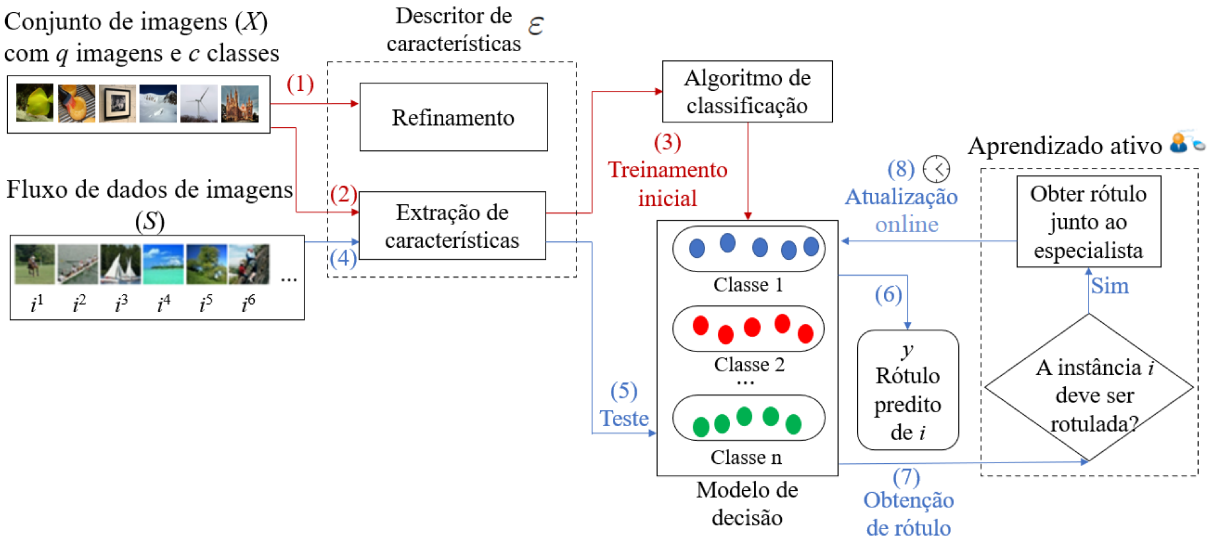

Figura 13 – Visão geral do método experimental considerado.

a extração das características das imagens de X a partir de ε. No caso da CNN, cada imagem do conjunto passa pela rede e como resultado é gerado um vetor ($\vec{\boldsymbol{x}}$) com d características para cada imagem. Os vetores de características podem ser utilizados como entrada do algoritmo de classificação de imagens.

A partir do conjunto de imagens X, que já passou pelo processo de extração de características, os vetores $\vec{\boldsymbol{x}}$ são utilizados para a construção inicial do modelo de decisão na Etapa (3) do método experimental. Após essa etapa, o fluxo de dados de imagens S é considerado. Na Etapa (4), a extração de características de cada imagem de S é realizada considerando ε. Vale destacar que ao longo do fluxo de dados S novas classes de imagens podem surgir, fazendo com que S possua uma quantidade de classes de imagens $c^{'}$. Além disso, a quantidade d de características das imagens pode não ser suficientemente representativa para a representação das novas imagens que surgem em S.

Com o surgimento de novas classes de imagens e novas instâncias de imagens com características diferentes em cada *timestamp* de S, pode ser necessária a evolução do descritor de características ε para considerar as novas características das imagens. Assim, um dos objetivos do método experimental descrito aqui é realizar diversas variações da quantidade de classes de imagens e do número de imagens do conjunto inicial X, para avaliar a influência desses parâmetros na representação do fluxo de dados de imagens S. A partir dessas variações, também pode ser possível observar a influência da descrição de imagens na qualidade do resultado da classificação de imagens.

Na Etapa (5) do método experimental, os vetores $\vec{\boldsymbol{x}}$ das imagens são utilizados para o teste do modelo de decisão. Como resultado, na Etapa (6), um rótulo y é predito pelo modelo de decisão para cada vetor $\vec{\boldsymbol{x}}$. Após essa etapa, verifica-se na Etapa (7), por meio de métodos de aprendizado ativo (veja Seção 2.3.2), se os rótulos das imagens devem ser informados por um usuário especialista. Esse processo também pode sofrer a interferência de latência (veja Seção 2.3.1.3). Finalmente, o treinamento online do modelo de decisão

é realizado na Etapa (8). A implementação dessas etapas do método experimental foi realizada por meio do *framework* EVISClass, detalhado na Seção 6.1.

Para a realização dos experimentos com o método experimental descrito foram selecionados 4 conjuntos de imagens reais, comumente, utilizados em trabalhos de classificação de imagens: *Scenes15*, *UIUC Sports*, *Flowers17* e *CIFAR10* (veja Seção 4.2).

O descritor CNN foi considerado para a extração de características das imagens. A arquitetura de rede selecionada para os experimentos foi a VGG16. A Etapa (1), do método experimental apresentado na Figura 13, utilizou a rede VGG16 pré-treinada no conjunto de imagens *ImageNet*. Um subconjunto de imagens da base de dados X, que contém q imagens de cada uma das c classes escolhidas, foi considerado para o refinamento da CNN. Os valores usados para q e c serão detalhados no Cenário 1 de experimentos. Nessa etapa de refinamento, os ajustes de pesos da rede foram realizados considerando 0 épocas com taxa de aprendizado igual a 0.1.

Os experimentos foram realizados considerando 3 diferentes cenários os quais foram contrastados com o resultado obtido por um caso base. Os cenários de experimentos propostos têm como objetivo analisar diferentes questões envolvidas na classificação de fluxos de dados de imagens, tais como: latência, *feature-evolution* e os fenômenos *concept-drift* e *concept-evolution*.

O Caso Base de experimentos, descrito a seguir, apresenta os resultados de uma análise comparativa de diferentes algoritmos para classificação de fluxos de imagens. Os resultados obtidos nos experimentos descritos nessa seção são utilizados como caso base para os demais experimentos. Os experimentos descritos no Cenário 1 exploram a utilização de diferentes quantidades de imagens para o refinamento do descritor de características. Os experimentos apresentados no Cenário 2 investigam a utilização de latência e comparam os resultados com o caso base. Técnicas de aprendizado ativo para a redução do número de instâncias com rótulos informados foram exploradas nos experimentos descritos no Cenário 3. Por fim, é apresentada uma análise estatística dos resultados experimentais obtidos.

6.2.2.1 Caso base - Avaliando diferentes algoritmos de fluxos de dados para classificação de fluxos de imagens

Os experimentos descritos aqui têm o objetivo de investigar a questão de pesquisa *Q1*: *"Estratégias de classificação probabilísticas ou baseadas em hiperplanos (como árvores de decisão, SVM e Naive Bayes) são melhores para classificar fluxos de imagens do que algoritmos baseados em técnicas de agrupamento, que são comumente usadas na literatura?"* Para tanto, a ferramenta MOA (*Massive Online Analysis*) (BIFET et al., 2018) foi considerada, devido a diversidade de algoritmos disponíveis na ferramenta. As mesmas condições disponíveis no *framework* EVISClass foram simuladas, tais como: utilização do método *Interleaved Test-Then-Train* para avaliação; treinamento inicial com um conjunto de imagens X, com q imagens e c classes.

Com o objetivo de avaliar diferentes estratégias de classificação, foram selecionados os seguintes algoritmos na ferramenta MOA: *Naïve Bayes* (D., 2016), *SGD Muticlass* (NANDI et al., 2021), *HoeffdingTree* (PFAHRINGER; HOLMES; KIRKBY, 2007) e *SAMKnn* (LOSING; HAMMER; WERSING, 2016). Esses algoritmos também foram utilizados em (PUGLIESE; COSTA; HIRATA, 2020; BIFET; FRANK, 2010) para a classificação de fluxos de dados. Para avaliar o desempenho desses algoritmos em relação à classificação de fluxos de dados de imagens, os resultados foram comparados em relação ao algoritmo NCM. Para o algoritmo *SAMKnn* foram usados os valores $K = 1$ e $K = 3$ para a vizinhança. A eficácia dos algoritmos foi avaliada pelas métricas acurácia e *F-Measure* (veja Seção 2.4.3). A Figura 14 apresenta uma visão geral da organização do conjunto de dados utilizado nesse experimento.

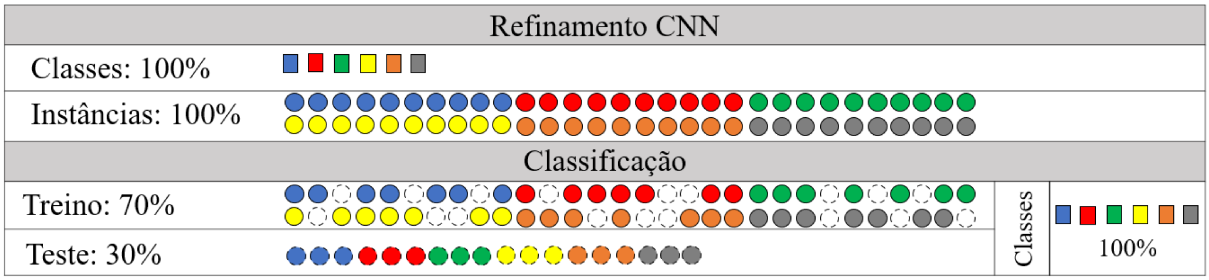

Figura 14 – Visão geral da organização dos conjuntos de dados no experimento para avaliar diferentes algoritmos de fluxos de dados para classificação de fluxos de imagens.

Conforme pode ser observado na Figura 14, o conjunto de imagens X utilizado para o refinamento da CNN (Etapa (1) da Figura 13) é composto por todas as imagens do conjunto de dados. 70% das imagens foram consideradas para o treinamento inicial do modelo de decisão (Etapa (3)). As demais imagens (30%) foram utilizadas para a composição do fluxo de dados S utilizado a partir da Etapa (4) do método experimental. Na Etapa (8) foram consideradas todas as imagens de teste para a atualização do modelo de decisão, considerando latência nula. As imagens selecionadas em X e S simulam o fenômeno *concept-drift*, por meio do parâmetro *instanceOption* do Algoritmo 14. Os resultados desse experimento estão descritos na Tabela 10. Os resultados com os algoritmos da ferramenta MOA no conjunto de dados *UIUC Sports* também estão ilustrados graficamente na Figura 15. Nesses gráficos é informada a acurácia em relação as instâncias de dados do fluxo de dados S.

Analisando os resultados apresentados na Tabela 10 é possível perceber que o algoritmo NCM apresentou os melhores resultados de *F-measure* e acurácia. Em relação aos demais algoritmos utilizados na ferramenta MOA, com os resultados ilustrados nos gráficos apresentados na Figura 15, percebe-se que o algoritmo *SGD Muticlass* apresentou resultados próximos ao algoritmo NCM, com diferença máxima de 1,1% de acurácia entre os resultados. Os algoritmos *Naïve Bayes* e *HoeffdingTree* sempre apresentaram

resultados abaixo de 22% para acurácia e também *F-measure* para todos os conjuntos de dados considerados. Já o algoritmo *SAMKnn* apresentou resultados que apresentaram diferenças que variaram entre 3,5% e 17,3% nos conjuntos de imagens *Scenes15*, *UIUC Sports* e *Flowers17* em relação ao algoritmo NCM. A maior diferença, com o 36,6%, foi constatada no conjunto de imagens *Cifar*, que é o conjunto com o maior número de imagens considerado nos experimentos.

Tabela 10 – Acurácia (A) e *F-Measure* (M). Avaliação de diferentes algoritmos de fluxos de dados para classificação de fluxos de imagens.

Conjunto Imagens	SAMKnn K=1		SAMKnn K=3		SGD		Naive Bayes		Hoeffding Tree		NCM	
	(A)	(M)	(A)	(M)	(A)	(M)	(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	64,6	68,5	62,4	70,2	79,0	80,4	5,8	6,5	7,3	7,9	**79,7**	**81,1**
Scenes15	77,9	76,0	77,5	75,5	80,3	**82,5**	6,3	7,4	9,4	10,2	**81,4**	81,5
Sports	73,4	73,1	71,9	75,6	84,4	85,4	11,1	11,9	13,2	13,8	**85,4**	**86,7**
Cifar	62,9	61,3	62,3	60,6	95,2	96,7	18,8	19,3	21,1	21,7	**96,1**	**97,3**

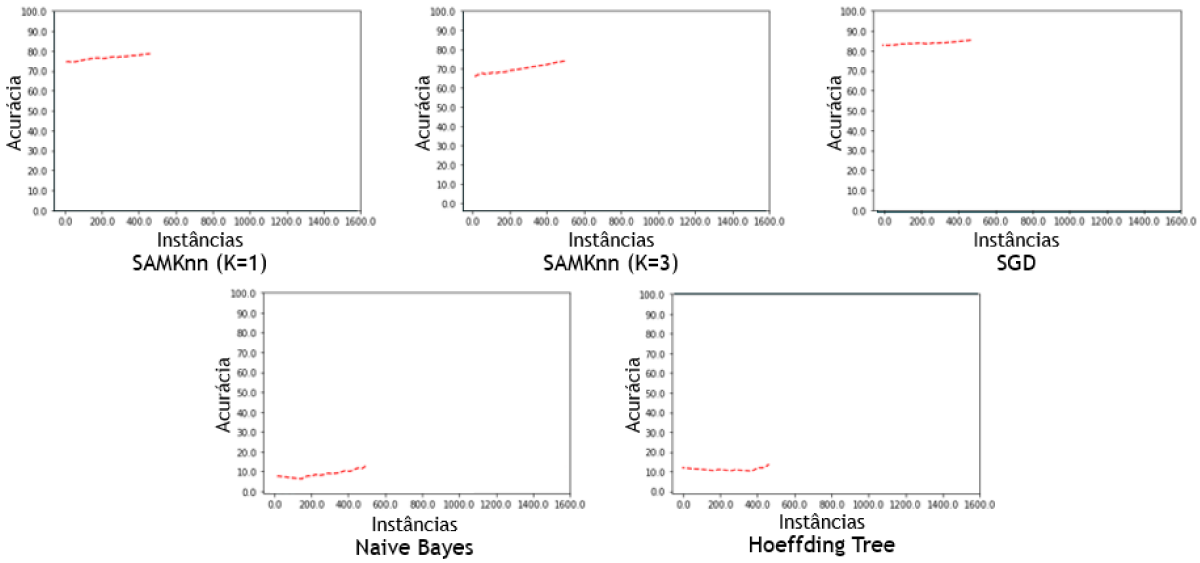

Figura 15 – Resultados (acurácia) dos algoritmos utilizados com a ferramenta MOA no conjunto de imagens *UIUC Sports*.

Com base nos resultados apresentados sobre a investigação da questão de pesquisa *Q1*, o algoritmo *SGD Muticlass* apresentou um potencial interessante para lidar com a classificação de fluxos de dados de imagens. Além disso, o algoritmo *SAMKnn* apresentou resultados interessantes nos conjuntos de dados *Scenes 15* e *UIUC Sports*. A Figura 15 evidencia esse comportamento no conjunto de dados *UIUC Sports*, mas também ilustra o comportamento geral dos algoritmos considerados nos demais conjuntos de dados. Dessa forma, esses algoritmos foram selecionados para compor a experimentação realizada nos próximos cenários, juntamente com o algoritmo NCM.

6.2.2.2 Cenário 1 - Avaliando diferentes conjuntos de imagens para o refinamento da CNN e o seu impacto no desempenho do classificador

Comumente a literatura tem assumido que todas as imagens do fluxo estão disponíveis para o refinamento da CNN. Como o fluxo é potencialmente infinito, essa suposição é irrealista. O que geralmente acontece é que um conjunto de imagens está disponível inicialmente para treinamento de um modelo de decisão inicial e esse mesmo conjunto pode ser usado para refinar a CNN. Dessa forma, os experimentos descritos aqui têm o objetivo de investigar a questão de pesquisa *Q2*: *"Qual o impacto do conjunto de dados usado no refinamento da CNN no desempenho do classificador"*. Para tanto, o experimento foi dividido em duas partes: redução do número de instâncias e redução simultânea do número de classes e também de instâncias.

Considerando a redução do número de imagens para o refinamento do descritor de características, esse cenário de experimentos também analisou o recurso de *feature-evolution*. Um algoritmo da literatura que trata este problema, o iCaRL (REBUFFI et al., 2017) foi considerado (veja Seção 3.1). Esse algoritmo possui um método descritor de características capaz de evoluir, isto é, além de realizar a classificação de fluxos de dados de imagens, esse algoritmo é capaz de trabalhar com *feature-evolution*. Dessa forma, esse algoritmo foi selecionado para contrastar os resultados em relação aos demais algoritmos, que não implementam *feature-evolution*. A implementação do algoritmo iCaRL, considerada nos experimentos apresentados aqui, foi a mesma utilizada em (REBUFFI et al., 2017).

Avaliando a redução do número de instâncias No experimento descrito aqui considera-se que todas as classes do problema são conhecidas e que apenas um subconjunto de instâncias de cada classe está disponível para o refinamento da CNN e treinamento do modelo inicial. O objetivo foi avaliar o impacto da redução do número de imagens do conjunto inicial (X), usado na Etapa (1) do método experimental considerado. A Figura 16 apresenta uma visão geral da organização do conjunto de dados utilizado nesse experimento, considerando 50% de instâncias. É importante destacar que o conjunto de teste é sempre formado pelas mesmas instâncias, que representam 30% da base de dados. O motivo para isso é garantir a comparação dos experimentos. Considerando o cenário caso base em que temos 70% das instâncias para treino e este cenário em que temos apenas 50% das instâncias para treino, significa que 20% dessas instâncias foram descartadas

Como pode ser observado na Figura 16, o número de classes de imagens utilizado em X foi preservado, ou seja, todas as classes foram consideradas. No cenário de experimentos descrito aqui o número de imagens de cada classe foi reduzido para 70%, 50% e 30% em relação ao caso base. Cada conjunto de imagens usado para compor X foi selecionado de forma a simular o fenômeno *concept-drift* (parâmetro *instanceOption* do framework EVISClass). Os resultados dos experimentos estão descritos na Tabela 11. A coluna Instâncias representa a quantidade de imagens utilizada em cada experimento.

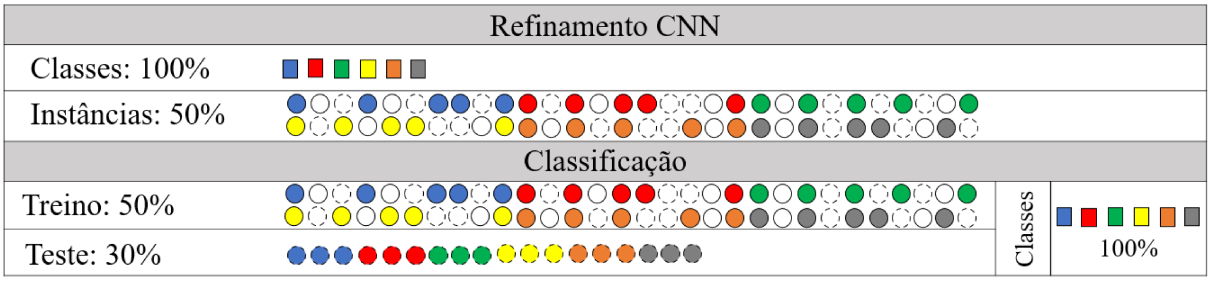

Figura 16 – Visão geral da organização dos conjuntos de dados considerando a redução no número de instâncias.

Observando a Tabela 11 é possível constatar que os resultados de acurácia e F-measure degradaram para todos os conjuntos de imagens. As degradações variaram entre 0,2% e 2,5%. A degradação máxima de 2,5% foi observada na acurácia do conjunto de imagens *Cifar* com o algoritmo *SAMKnn*. Quando se compara os resultados da Tabela 11 com o caso base, presente na Tabela 10, é possível observar ainda maiores degradações. Por exemplo, analisando os resultados obtidos para os algoritmos *SAMKnn* e *SGD*, no conjunto de imagens *Cifar*, foram observadas degradações de 2,9% e 3,2%, respectivamente.

Tabela 11 – Acurácia (A) e *F-Measure* (M). Avaliação da redução do número de instâncias

Conjunto Imagens	Instâncias	SAMKnn K=1 (A)	SAMKnn K=1 (M)	SAMKnn K=3 (A)	SAMKnn K=3 (M)	SGD (A)	SGD (M)	NCM (A)	NCM (M)	iCaRL (A)	iCaRL (M)
Flowers17	70%	**64,5**	**68,4**	**62,1**	**69,4**	**78,5**	**79,6**	**79,6**	**80,3**	**79,3**	**80,7**
	50%	64,2	68,3	61,4	68,6	77,6	78,8	79,0	79,8	79,0	80,4
	30%	63,3	67,7	60,5	68,5	77,5	77,8	78,5	79,5	78,9	80,3
Scenes15	70%	**77,6**	**75,6**	**76,8**	**74,6**	**79,6**	**82,3**	**80,4**	**80,5**	**80,9**	**80,8**
	50%	77,4	75,2	76,6	73,6	79,4	81,3	79,7	80,4	80,6	80,5
	30%	76,4	74,2	75,8	73,3	78,4	81,1	79,0	80,1	80,3	80,3
Sports	70%	**73,3**	**72,4**	**71,5**	**75,5**	**84,2**	**85,3**	**84,9**	**86,4**	**85,3**	**86,4**
	50%	73,2	72,0	71,4	74,8	83,6	84,6	84,2	86,2	85,0	86,3
	30%	72,7	71,5	70,7	74,1	83,3	84,4	83,8	85,5	84,8	86,2
Cifar	70%	**62,5**	**59,6**	**61,7**	**59,8**	**94,7**	**96,6**	**95,6**	**96,4**	**95,8**	**96,7**
	50%	60,8	59,0	60,8	58,8	94,1	95,1	94,6	95,7	95,6	96,5
	30%	60,0	58,1	59,4	57,9	93,7	94,9	93,7	94,2	95,5	96,3

Em relação ao algoritmo *iCaRL*, que considera *feature-evolution*, no geral observou-se os melhores resultados, com as menores degradações. A degradação máxima observada foi de 0,6% na acurácia do conjunto de imagens *Scenes 15*. Esses resultados mostram que a redução do número de imagens para o refinamento dos descritores pode prejudicar a capacidade de representação de todo o fluxo de dados de imagens e, consequentemente, o desempenho do classificador.

Avaliando a redução simultânea do número de classes e de instâncias No experimento descrito aqui foi considerado que um subconjunto das classes do problema está disponível para refinamento da CNN e também um subconjunto de instâncias de cada classe. Portanto, neste cenário de experimentos foi avaliada a junção dos cenários anteriores, isto é, a redução simultânea do número de classes de imagens e da quantidade de imagens. A Figura 17 apresenta uma visão geral da organização do conjunto de dados utilizado nesse experimento, considerando 50% de classes e 50% de instâncias de cada classe.

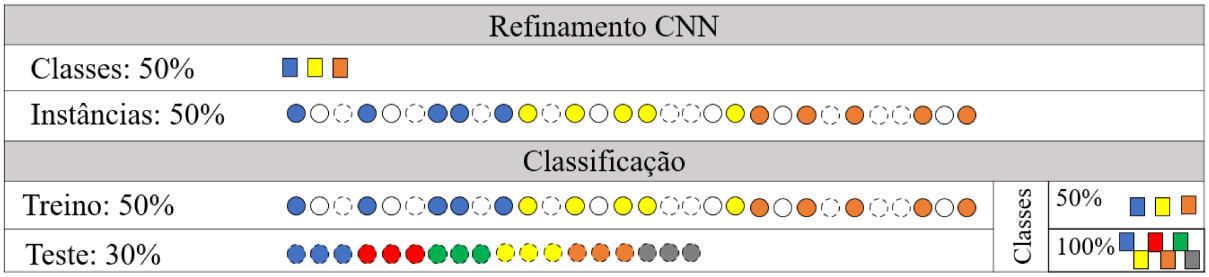

Figura 17 – Visão geral da organização dos conjuntos de dados considerando a redução simultânea no número de classes e no número de instâncias.

Quando se compara os cenários dos experimentos anteriores com o cenário ilustrado na Figura 17, é possível notar que o conjunto inicial (X), empregado na Etapa (1) do método experimental considerado, foi construído com a menor quantidade de instâncias, já que também ocorre a redução no número de classes. Nesse experimento, a quantidade de classes utilizada foi reduzida para 50% e 30%. Para o número de imagens selecionadas foram consideradas as mesmas porcentagens (50% e 30%). Os resultados desse experimento estão apresentados na Tabela 12.

Observando a Tabela 12 foi possível constatar que existe uma degradação gradativa dos resultados a cada redução de classes e também de imagens. Considerando somente a redução da quantidade de imagens já é possível observar degradações nos resultados. Entretanto, os menores resultados observados foram obtidos para os casos de 30% de classes e 30% de imagens. Comparando a acurácia dos casos extremos da Tabela 12, isto é, 50% classes com 50% imagens e 30% classes com 30% imagens, a maior degradação observada foi de 7,0% para o conjunto *Flowers17* no algoritmo *SAMKnn*. O algoritmo *iCaRL* apresentou as menores degradações, com o valor máximo observado de 1,2%. Em relação ao caso base, descrito na Tabela 10, foram observadas as maiores degradações em relação a todos os cenários de experimentos. O maior valor observado foi de 10,0%, no algoritmo *SAMKnn* para o conjunto *Scenes15*.

Analisando esses resultados é possível constatar, que um algoritmo que considera *feature-evolution* pode amenizar os efeitos da redução do número de classes e do número de imagens. Entretanto, nos demais algoritmos, essas reduções provocaram uma degradação ainda maior dos resultados quando comparado aos cenários de experimentos anteriores.

Tabela 12 – Acurácia (A) e *F-Measure* (M). Avaliação da redução do número de classes e do número de instâncias.

			SAMKnn				SGD		NCM		iCaRL	
Conjunto Imagens	Classes	Instâncias	K=1		K=3							
			(A)	(M)	(A)	(M)	(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	50%	50%	**62,3**	**66,2**	**61,8**	**67,7**	**76,5**	**76,5**	**77,6**	**78,0**	**79,0**	**80,5**
		30%	60,5	63,7	58,8	65,3	75,2	74,1	75,9	77,1	78,8	80,2
	30%	50%	58,5	63,0	57,2	65,2	73,3	71,2	74,1	77,0	78,5	80,1
		30%	57,0	60,0	54,8	63,9	70,8	71,0	73,6	75,7	78,2	80,0
Scenes15	50%	50%	**76,6**	**72,5**	**76,6**	**74,5**	**78,2**	**81,5**	**79,7**	**80,3**	**80,7**	**80,5**
		30%	75,2	71,5	74,6	74,4	77,8	79,2	77,2	78,5	80,6	80,4
	30%	50%	73,4	68,8	72,7	73,5	75,9	76,4	76,1	77,0	80,4	80,2
		30%	72,8	66,0	70,5	72,2	74,7	75,2	74,5	76,8	80,3	80,1
Sports	50%	50%	**72,6**	**69,6**	**68,2**	**72,4**	**81,3**	**82,2**	**82,1**	**84,7**	**84,2**	**85,8**
		30%	71,4	68,0	67,4	71,3	80,7	80,5	80,1	83,0	84,1	85,7
	30%	50%	70,0	67,9	66,2	70,8	79,4	79,1	79,7	82,4	83,9	85,4
		30%	68,4	67,8	64,9	69,6	78,7	78,9	79,0	80,7	83,6	85,1
Cifar	50%	50%	**59,7**	**57,5**	**59,3**	**56,5**	**92,7**	**93,6**	**94,0**	**95,4**	**95,7**	**96,4**
		30%	58,0	56,0	58,4	55,1	92,0	93,3	93,2	94,6	95,5	96,2
	30%	50%	56,5	53,3	57,2	53,7	91,9	93,1	90,6	94,0	95,1	96,1
		30%	54,1	52,2	56,5	53,6	90,3	92,2	88,3	91,4	94,5	96,0

Então, para algoritmos que não consideram *feature-evolution*, as quantidades de imagens e de classes de imagens consideradas para o refinamento do descritor CNN e no treinamento inicial dos classificadores, podem influenciar nos resultados obtidos.

6.2.2.3 Cenário 2 - Avaliando o impacto da latência no desempenho dos classificadores

O experimento descrito aqui tem o objetivo de investigar a questão de pesquisa *Q3*: *"Qual é o impacto da latência na eficácia de classificadores de fluxos de dados de imagens?"*. Nesse conjunto de experimentos avaliou-se o impacto do atraso na informação dos rótulos das imagens para atualização do modelo no desempenho do classificador. Diferentes valores de latência foram considerados na realização dos experimentos (parâmetro l do framework EVISClass). Dois cenários foram considerados para a organização dos conjuntos de dados de imagens: um cenário com todas as classes de imagens disponíveis no treinamento inicial e outro cenário com somente um subconjunto de classes disponível no treinamento inicial.

Considerando todas as classes de imagens no treinamento inicial A mesma organização do conjunto de dados utilizada no caso base, ilustrada na Figura 14, foi usada

neste cenário de experimentos, para o refinamento da CNN e para o treinamento inicial dos classificadores. O objetivo desse experimento consiste em avaliar a latência em ambientes que ocorrem *concept-drift*, mas sem a presença do fenômeno *concept-evolution*, ou seja, todas as classes de imagens são conhecidas no treinamento inicial. Os resultados dos experimentos são apresentados na Tabela 13. Os valores da segunda coluna referem-se a latência utilizada. Os resultados com latência nula (valor 0) são os mesmos presentes na Tabela 10.

Tabela 13 – Acurácia (A) e *F-Measure* (M). Avaliação do impacto da latência no desempenho dos classificadores - todas as classes de imagens no treinamento inicial

		SAMKnn				SGD		NCM	
Conjunto Imagens	Latência	K=1		K=3					
		(A)	(M)	(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	0	**64,6**	**68,5**	**62,4**	**70,2**	**79,0**	**80,4**	**79,7**	**81,1**
	10	63,6	68,1	61,8	69,4	78,3	79,5	79,0	80,4
	100	63,4	67,4	59,9	67,2	76,9	78,8	78,9	79,7
	1000	62,8	64,5	58,5	66,4	74,2	78,1	77,3	77,5
Scenes15	0	**77,9**	**76,0**	**77,5**	**75,5**	**80,3**	**82,5**	**81,4**	**81,5**
	10	77,5	75,9	76,8	75,4	79,4	81,5	80,5	80,8
	100	75,7	73,6	76,2	73,0	77,4	81,1	78,5	79,6
	1000	74,0	71,3	75,2	72,8	74,4	80,9	77,9	78,8
Sports	0	**73,4**	**73,1**	**71,9**	**75,6**	**84,4**	**85,4**	**85,4**	**86,7**
	10	73,2	72,8	71,2	74,9	84,2	84,9	84,4	85,9
	100	70,6	70,7	69,1	73,0	82,1	82,0	82,1	85,6
	1000	69,6	70,4	68,0	71,0	81,5	81,1	82,0	83,9
Cifar	0	**62,9**	**61,3**	**62,3**	**60,6**	**95,2**	**96,7**	**96,1**	**97,3**
	10	62,4	60,6	61,5	59,6	94,8	96,5	95,7	96,4
	100	61,7	60,1	60,2	59,2	93,2	96,0	95,6	94,1
	1000	59,3	60,0	58,1	58,5	91,7	93,9	93,6	92,5

Analisando a Tabela 13 fica evidente que quanto maior o valor da latência maior é a degradação do desempenho dos classificadores. Em todos os casos observados, os menores valores obtidos de acurácia e *F-measure* foram para a latência 1.000. A maior degradação registrada entre os valores de latência 0 e 1.000 foi de 5,9%, observada na acurácia do algoritmo SGD, para o conjunto de dados *Scenes15*.

Considerando um subconjunto de classes de imagens no treinamento inicial Esse cenário de experimentos considerou um subconjunto de classes tanto para o refinamento da CNN quanto para o treinamento inicial dos classificadores. A mesma organização do conjunto de dados ilustrada na Figura 17 foi usada. O objetivo desse

experimento consiste em avaliar a latência em ambientes com a presença dos fenômenos *concept-evolution* e *concept-drift*. Os resultados dos experimentos estão na Tabela 14. Os resultados com latência nula (valor 0) são os mesmos presentes na Tabela 12, com 30% de classes e 30% de instâncias.

Tabela 14 – Acurácia (A) e *F-Measure* (M). Avaliação do impacto da latência no desempenho dos classificadores - subconjunto de classes de imagens no treinamento inicial

		SAMKnn				SGD		NCM	
Conjunto Imagens	Latência	K=1		K=3					
		(A)	(M)	(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	0	**57,0**	**60,0**	**54,8**	**63,9**	**70,8**	**71,0**	**73,6**	**75,7**
	10	51,9	56,3	51,1	58,5	66,7	66,3	68,3	71,0
	100	28,3	32,8	27,2	35,2	44,3	43,7	45,6	47,0
	1000	3,3	3,4	3,7	3,2	8,6	8,4	9,7	10,3
Scenes15	0	**72,8**	**66,0**	**70,5**	**72,2**	**74,7**	**75,2**	**74,5**	**76,8**
	10	66,9	60,5	66,9	69,0	70,9	69,9	68,7	73,4
	100	44,3	39,1	43,5	47,5	49,1	46,7	46,6	49,2
	1000	5,6	5,3	5,7	5,5	16,6	15,7	16,9	16,5
Sports	0	**68,4**	**67,8**	**64,9**	**69,6**	**78,7**	**78,9**	**79,0**	**80,7**
	10	65,1	62,7	59,9	66,0	73,5	72,9	73,4	76,2
	100	42,1	40,6	37,7	40,2	51,3	49,9	50,4	52,0
	1000	3,5	3,9	3,6	3,9	9,8	10,2	11,7	11,8
Cifar	0	**54,1**	**52,2**	**56,5**	**53,6**	**90,3**	**92,2**	**88,3**	**91,4**
	10	49,4	47,9	53,1	50,6	84,7	86,2	82,4	87,4
	100	28,3	27,1	33,0	27,7	60,4	62,2	59,4	62,7
	1000	4,1	6,1	8,7	7,6	19,4	21,6	21,9	22,8

A partir da análise da Tabela 14 é possível observar o mesmo comportamento do cenário anterior. Quanto maior o valor da latência, maior é a degradação do desempenho dos classificadores. Além disso, quando se compara os resultados apresentados pelos valores de latência 0 e 1.000 observa-se degradações de até 70,9%. Isso demonstra que a ocorrência do fenômeno *concept-evolution* é ainda mais significativa para os cenários com valores elevados de latência. Em especial, para os conjuntos de imagens *Flowers17* e *Sports*, que possuem as menores quantidades de instâncias, os valores de acurácia e *F-measure* foram os menores registrados, o que mostra que a atualização online do modelo de decisão em tempo hábil é importante para lidar com os fenômenos *concept-drift* e *concept-evolution*.

6.2.2.4 Cenário 3 - Avaliando o impacto de estratégias de aprendizado ativo no desempenho de classificadores

O objetivo do último cenário de experimentos foi investigar a questão de pesquisa *Q4*: *"Qual é o impacto da quantidade de rótulos informada para a atualização do modelo de decisão na eficácia de classificadores de fluxos de dados de imagens?"*. Para tanto, duas estratégias de informação de rótulos foram consideradas (parâmetro *labelOption* do framework EVISClass): definir aleatoriamente as imagens que terão os rótulos informados e selecionar as imagens baseado no grau de incerteza da classificação. As mesmas configurações dos conjuntos de dados utilizados na seção anterior foram consideradas para a realização do experimento descrito aqui, com 30% de classes e 30% de instâncias no treinamento inicial do classificador. Além disso, para cada uma das estratégias de informação de rótulos foram considerados diferentes valores de latência. Os resultados obtidos no cenário de experimentos descrito aqui estão apresentados na Tabela 15.

Analisando esses resultados é possível constatar que quanto maior a latência, maior a degradação dos valores de acurácia e *F-measure*, independente da estratégia de informação de rótulos considerada. Para a estratégia aleatória foram informados, em média, 50% dos rótulos das instâncias, já para a estratégia baseada em incerteza foi cerca de 40%. Então, fica evidente que uma estratégia mais sofisticada de seleção de instâncias, como é o caso da abordagem baseada em incerteza, é capaz de produzir valores superiores de acurácia e *F-measure*, mesmo com uma menor quantidade de rótulos informada. Quando se compara os resultados apresentados na Tabela 14 com o experimento descrito aqui, desconsiderando a latência 1.000, observa-se degradações de até 11,5% na estratégia aleatória e de até 9,8% na estratégia baseada em incerteza. De forma a complementar a análise dos resultados, foi realizado o teste estatístico descrito na Seção 6.2.2.5.

6.2.2.5 Análise Estatística

A análise estatística realizada comparou os diferentes cenários de experimentos realizados, considerando os valores de acurácia e *F-measure*. Foi aplicado o *Wilcoxon signed-rank test* (veja Seção 4.3.2). A hipótese nula formulada foi: H_0 − os desempenhos preditivos apresentados em dois cenários de experimentos (A e B) não são estatisticamente diferentes, a um nível de confiança de 95%. A hipótese alternativa formulada foi: H_1 − o desempenho preditivo apresentado no cenário de experimentos A é superior ao desempenho preditivo apresentado no cenário de experimentos B.

A primeira análise realizada comparou os resultados obtidos pelo Caso Base (Tabela 10) com os resultados obtidos pelos demais cenários de experimentos (Tabelas 11,12, 13, 14 e 15). Em todos os casos os resultados se mostraram estatisticamente diferentes, ou seja, H0 foi rejeitada. Sendo possível concluir que os resultados da Tabela 10 foram superiores em relação aos demais resultados comparados.

Tabela 15 – Acurácia (A) e *F-Measure* (M). Avaliação do impacto de estratégias de aprendizado ativo no desempenho de classificadores.

			SAMKnn				SGD		NCM	
Conjunto Imagens	Estratégia Rótulos	Latência	K=1		K=3					
			(A)	(M)	(A)	(M)	(A)	(M)	(A)	(M)
Flowers17	Aleatória	0	47,7	50,2	45,9	55,3	61,1	62,2	64,8	67,2
		10	42,2	47,3	40,5	47,7	56,6	55,8	58,0	61,9
		100	16,8	22,4	16,5	23,9	33,1	33,2	34,5	37,0
		1000	0,2	0,4	0,3	0,2	4,6	4,3	6,7	6,8
	Incerteza	0	**52,6**	**54,6**	**50,5**	**60,0**	**65,9**	**67,0**	**68,8**	**71,9**
		10	45,6	50,9	43,8	50,8	60,2	59,2	61,8	65,8
		100	18,8	24,6	18,8	26,0	35,5	36,0	36,5	39,0
		1000	1,3	1,9	1,4	1,6	6,0	5,7	7,7	8,3
Scenes15	Aleatória	0	64,9	59,0	63,3	64,5	67,5	68,2	66,6	68,8
		10	58,5	52,3	58,5	60,2	62,2	61,3	63,2	62,2
		100	34,9	29,3	34,2	38,2	39,4	36,7	39,0	41,8
		1000	1,3	1,1	1,6	1,4	8,3	8,5	9,1	9,5
	Incerteza	0	**70,8**	**64,4**	**68,5**	**69,5**	**72,6**	**73,3**	**71,7**	**74,3**
		10	62,7	57,1	62,6	64,4	67,0	65,3	67,6	66,9
		100	39,7	33,9	38,7	42,5	44,3	40,9	43,4	46,8
		1000	2,3	2,5	2,6	2,8	9,5	9,5	10,3	10,8
Sports	Aleatória	0	60,1	58,0	55,0	59,6	70,6	68,9	70,7	72,6
		10	54,4	51,7	49,2	55,5	64,2	63,0	64,4	66,6
		100	31,4	30,1	27,3	29,7	39,8	38,9	39,5	41,6
		1000	0,1	0,2	0,2	0,2	2,6	2,2	4,7	4,5
	Incerteza	0	**65,0**	**62,8**	**59,4**	**64,5**	**74,7**	**73,6**	**74,9**	**76,6**
		10	57,8	55,1	53,0	58,8	67,5	67,0	67,6	70,3
		100	34,1	32,5	30,1	31,9	42,4	41,4	41,6	44,0
		1000	1,6	1,7	1,6	1,2	4,0	3,5	6,2	6,0
Cifar	Aleatória	0	46,6	44,9	49,2	46,3	83,3	85,0	84,3	85,2
		10	40,8	39,5	44,3	42,6	75,8	77,5	77,1	78,8
		100	19,3	17,1	23,8	18,1	51,0	52,7	51,8	53,3
		1000	2,2	2,1	2,3	2,2	9,4	9,3	12,3	12,5
	Incerteza	0	**51,6**	**49,9**	**55,1**	**51,4**	**87,8**	**90,4**	**88,1**	**90,5**
		10	44,8	44,0	49,2	47,0	80,8	81,6	81,8	82,8
		100	23,7	21,7	28,7	22,1	55,0	57,7	55,9	57,3
		1000	3,7	3,6	3,7	3,6	10,4	10,5	13,7	13,8

A segunda análise estatística realizada considerou os resultados obtidos por cada um dos experimentos realizados para abordar a questão de pesquisa *Q2* (Tabela 11 e Tabela 12). Para tanto, os resultados com 70% de instâncias foram comparados em relação aos demais valores (50% e 30%). Em todas as comparações os resultados se mostraram estatisticamente diferentes, sendo possível concluir que os resultados com 70% de instâncias foram superiores em relação as demais configurações consideradas.

A terceira análise estatística também considerou os resultados obtidos por cada um dos experimentos realizados para abordar a questão de pesquisa *Q2* (Tabela 11 e Tabela 12), mas com um objetivo diferente. Nessa análise estatística o objetivo foi avaliar os resultados do algoritmo iCaRL em relação aos demais algoritmos. Em todas as análises os resultados do algoritmo iCaRL se mostraram estatisticamente diferentes, ou seja, os resultados do algoritmo iCaRL10 foram superiores em relação aos demais algoritmos comparados.

Na quarta análise estatística os resultados obtidos pelos experimentos que consideraram latência foram avaliados (questão de pesquisa *Q3*). Para tanto, os resultados experimentais apresentados nas Tabelas 13 e 14 foram comparados. Os resultados apresentados na Tabela 13 foram superiores aos resultados da Tabela 14, o que demonstra que a ocorrência do fenômeno *concept-evolution* (Tabela 14) pode interferir significativamente nos cenários de latência.

Por fim, a última análise estatística comparou os resultados obtidos pelos experimentos sobre as estratégias de aprendizado ativo (questão de pesquisa *Q4*, veja Tabela 15). Os resultados apresentados pela estratégia baseada em incerteza foram superiores em relação aos resultados da estratégia aleatória, o que pode demonstrar que com estratégias eficientes para a informação de rótulos, mesmo com uma quantidade inferior de rótulos informados, é possível obter resultados interessantes.

6.3 Considerações Finais

A avaliação de classificadores de fluxos de imagens é uma questão de pesquisa importante ainda não solucionada pela literatura científica da área. Os trabalhos correlatos existentes não consideram cenários de avaliação semelhantes aos cenários de aplicações reais. Com a análise dos métodos empregados na avaliação de classificadores de fluxos de imagens nos trabalhos da literatura da área foram identificadas as limitações dos métodos existentes para a avaliação de cenários de fluxos de imagens reais. Para lidar com essas limitações foi desenvolvido o *framework* EVISClass que permite a avaliação de classificadores de fluxos de imagens considerando todos os seguintes aspectos: atualização online do modelo de decisão, *concept-evolution*, *concept-drift* e a ausência de rótulos para as imagens. Analisando os resultados experimentais obtidos foi possível constatar que quanto maior a realidade considerada nos métodos de avaliação, maior a degradação dos

resultados do classificador avaliado. Além disso, a estratégia de fornecimento de rótulos adotada possui forte influência na acurácia obtida.

Este capítulo também realizou uma análise experimental, utilizando o *framework* EVISClass, sobre a refinamento do descritor de características CNN para a classificação de fluxos de dados de imagem. A partir dessa análise foi constatado que um fator que pode interferir para a diminuição da acurácia nos resultados da classificação é a quantidade de imagens utilizada na fase de refinamento do descritor de características. O cenário no qual foram observadas as maiores degradações dos resultados foi o que usou maiores valores de latência e as menores quantidades de imagens para o refinamento do descritor de características. Também foi possível observar que um algoritmo que considera *feature-evolution* pode ser capaz de minimizar as degradações dos resultados. Outra questão sobre a atualização do modelo de decisão é que utilizando métodos de aprendizado ativo é possível atualizar o modelo com uma menor quantidade de imagens, sem grande impacto na qualidade dos resultados.

Capítulo 7

Algoritmo baseado em *hubs* para a classificação de fluxo de dados de imagens

Este capítulo apresenta o algoritmo HubISC *(Hubness for Image data Stream Classification)*, que considera o uso de *hubs*, para a classificação de fluxos de dados de imagens. O aspecto *hubness* é uma propriedade inerente de conjuntos de dados de alta dimensão. Esse aspecto já foi aplicado com eficácia na classificação de imagens em cenários estáticos e não foi explorado para a classificação de fluxos de dados de imagens. *Hubs* são instâncias de dados que aparecem na lista de vizinhos mais próximos de um grande número de outras instâncias (veja Seção 2.2). Os *hubs*, geralmente, estão localizados em regiões densas. Então, podem ser usados como representantes de classes, de maneira similar a como ocorre com centróides ou medóides. Além disso, os *hubs* podem indicar instâncias a serem rotuladas em processos de aprendizado ativo. Os detalhes do algoritmo HubISC são descritos na Seção 7.1. A Seção 7.2 descreve a avaliação experimental realizada com o algoritmo HubISC. Por fim, a Seção 7.3 apresenta as considerações finais deste capítulo.

7.1 HubISC

O algoritmo de classificação de fluxos de dados de imagens apresentado nesta tese tem como objetivo incorporar o uso de *hubs* (veja Seção 2.2) para a realização dessa tarefa. Esse algoritmo também tem o objetivo de tratar aspectos importantes nesse contexto, como: *I)* surgimento de novas classes (*concept-evolution*), isto é, na etapa de testes surgem novas classes diferentes das disponíveis na fase de treinamento; *II)* evolução dos conceitos de classes já existentes (*concept-drift*); *III)* utilização de diferentes quantidades de imagens rotuladas para a atualização do modelo de decisão. Para tanto, o algoritmo HubISC possui duas fases: fase *offline* para a construção inicial do modelo de decisão; fase *online* para a

classificação do fluxo de dados de imagens. O Algoritmo 15 apresenta uma visão geral do algoritmo HubISC.

Algoritmo 15: HubISC

Entrada: X conjunto de imagens rotulado, K vizinhança para cálculo dos *hubs*, h_K limiar *hubness*, S fluxo de imagens, L limiar de distância, T tamanho do *buffer*
Saída: desempenho preditivo do modelo de classificação f e quantidade de rótulos solicitados

```
1  início
       /* Fase offline                                                    */
2      X' ← extrai as características de cada imagem i ∈ X e separa as instâncias de X de acordo
       com a classe;
3      H ← identifica os hubs de X' a partir de em K e h_K;
4      f ← constrói o modelo de decisão inicial a partir de H;
       /* Fase online                                                     */
5      Z ← ∅;
6      repita
7          recebe i por meio de S;
8          x_i ← extrai as características de i;
           /* x_i é avaliado por f                                        */
9          se a distância de x_i em relação ao hub h mais próximo ≤ L então
10             y de x_i ← y de h;
11         senão
12         Z ← Z ∪ x_i;
13         se |Z| == T então
14             H' ← identifica os hubs h' ∈ Z considerando K e h_K;
15             P ← executa a estratégia de aprendizado considerando H';
               /* atualiza o modelo de decisão e executa estratégias de
                  esquecimento de representantes                          */
16             H ← H ∪ P;
17             atualiza H com estratégias de esquecimento de representantes;
18             Z ← ∅;
19     até o fim de S ;
20 fim
```

7.1.1 Fase *offline*

Na fase *offline*, o algoritmo HubISC recebe como parâmetros: um conjunto inicial de imagens X rotulado, com c classes e q imagens para cada classe; um valor K para o cálculo dos vizinhos mais próximos; um limiar h_K que determina a pontuação *hubness* mínima para uma instância ser considera *hub*. Cada imagem $i \in X$ deve ser representada por um vetor $\vec{\boldsymbol{x}}$, após passar por um método de extração de características.

Observa-se na linha 2 do Algoritmo 15, que o conjunto de imagens é organizado por classes. Na linha 3, cada imagem $i \in X^{'}$, representada pelo vetor $\vec{\boldsymbol{x}_i}$, terá a sua respectiva pontuação *hubness* calculada a partir do valor K. A partir desse cálculo, ocorre a identificação dos *hubs*. Caso a pontuação *hubness* da respectiva imagem seja superior ao parâmetro h_K, a imagem i é considerada um *hub*. Cada classe de imagem pode ter mais de um *hub* associado. Na linha 4 do Algoritmo 15, o modelo de decisão inicial f do algoritmo HubISC é composto por um conjunto H de *hubs* identificados. Somente os *hubs* e seus

respectivos rótulos são armazenados para a utilização nas demais etapas da fase *online*, as demais instâncias processadas na fase *offline* são descartadas pelo algoritmo. A Figura 18 ilustra os procedimentos realizados na fase *offline* do algoritmo HubISC. A Figura 18(a) representa o conjunto de imagens rotulado X, informado como parâmetro do algoritmo. O conjunto X possui 3 classes de imagens, ilustradas com cores distintas. Na Figura 18(b) ocorre o cálculo da pontuação *hubness* e a identificação dos *hubs* (representados em formato de triângulo) que formam o conjunto H.

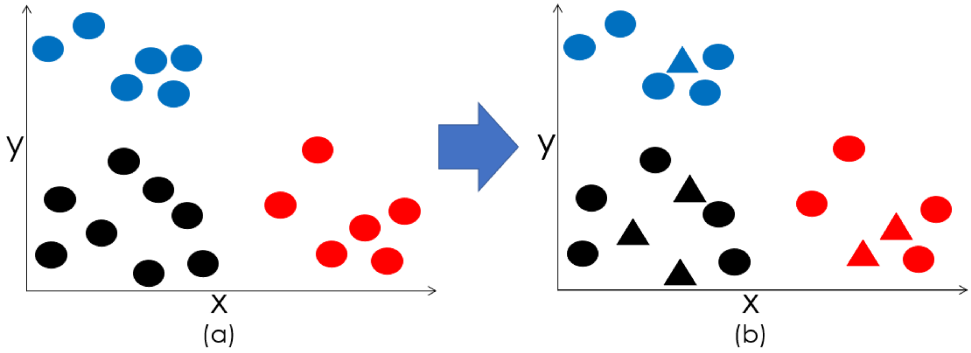

Figura 18 – Fase *offline* do algoritmo HubISC. (a) organização do conjunto de dados. (b) identificação dos *hubs*.

7.1.2 Fase *online*

Na fase *online*, ilustrada pela Figura 19, o algoritmo HubISC também considera os valores K e h_K da fase *offline*. Além disso, recebe como parâmetros: um fluxo de imagens S; um limiar L para determinar a distância D máxima permitida de uma instância i em relação aos *hubs*; um tamanho T para o *buffer* de instâncias. Cada imagem $i \in S$ também passará por um método de extração de características e será representada por um vetor $\vec{\boldsymbol{x}}$.

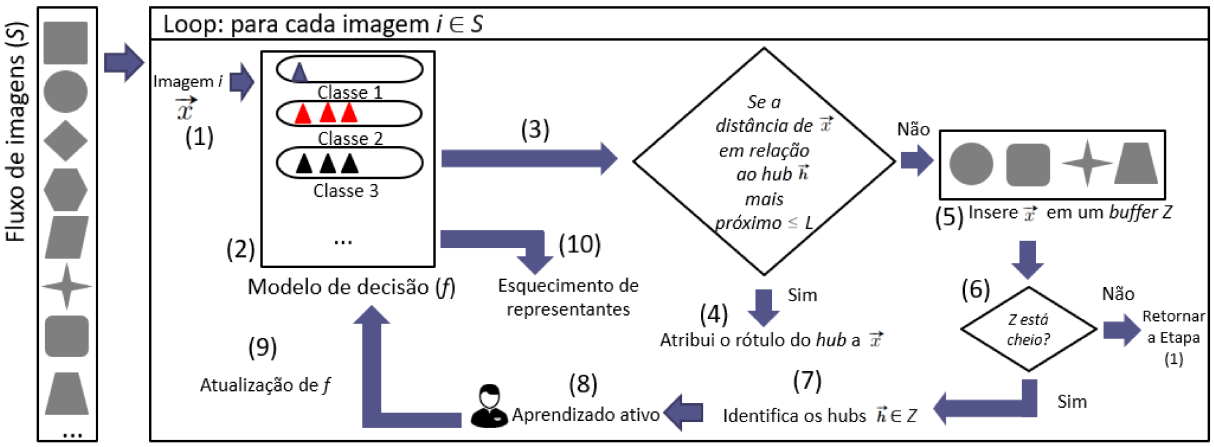

Figura 19 – Fase *online* do algoritmo HubISC.

Na Etapa (1) da fase *online* do algoritmo HubISC, cada imagem $i \in S$, representada por um vetor de características $\vec{\boldsymbol{x}}$, é avaliada pelo modelo de decisão f (linha 9 do Algoritmo

15). Na Etapa (3), é calculada a distância D de $\vec{\boldsymbol{x}}$ para cada *hub* $\vec{\boldsymbol{h}} \in H$. Então, o menor valor de D é analisado em relação ao limiar L. Caso $D \leq L$, o rótulo y do respectivo *hub* é atribuído a $\vec{\boldsymbol{x}}$ na Etapa (4) (linha 10 do Algoritmo 15). Senão, $\vec{\boldsymbol{x}}$ é inserida no *buffer* Z na Etapa (5) (linha 12 do Algoritmo 15).

O *buffer* Z possui um tamanho T predefinido. Na Etapa (6) da Figura 19, é verificado se o *buffer* está cheio, isto é, $|Z| == T$. Caso estiver, é calculada a pontuação *hubness* de cada vetor $\vec{\boldsymbol{x}} \in Z$ na Etapa (7) (linha 14 do Algoritmo 15). Dessa forma, se a pontuação *hubness* do vetor for superior a h_K, o respectivo vetor $\vec{\boldsymbol{x}}$ é considerado um *hub*. Os *hubs* identificados são armazenados em $H^{'}$.

Na Etapa (8) ocorre a fase de aprendizado ativo do algoritmo. Nessa etapa, são selecionados os *hubs* com maior pontuação *hubness* para a obtenção do respectivo rótulo desses hubs. Em um cenário de aplicações reais poderia considerar a interação com um usuário especialista nessa etapa. Então, o rótulo informado de cada *hub* é propagado para as demais instâncias. Para tanto, avalia-se a distância de cada vetor $\vec{\boldsymbol{x}} \in (Z - H^{'})$ para os *hubs* $\vec{\boldsymbol{h}}^{'} \in H^{'}$. Cada vetor $\vec{\boldsymbol{x}}$ recebe o mesmo rótulo do *hub* $\vec{\boldsymbol{h}}^{'}$ mais próximo (linha 15 do Algoritmo 15).

Observa-se que na Etapa (8) do algoritmo HubISC, com as instâncias rotuladas, pode ocorrer a identificação de novas classes e também a mudança de conceito das classes já conhecidas, sendo possível tratar os fenômenos *concept-evolution* e *concept-drift*. Na Etapa (9) do algoritmo ocorre a atualização do modelo de decisão f, para a atualização do conjunto H de *hubs*, com os novos *hubs* identificados na etapa anterior (linha 16 do Algoritmo 15). Finalmente, na Etapa (10) são executados os mecanismos de esquecimento de representantes (linha 17 do Algoritmo 15). O método de esquecimento implementado no HubISC está descrito na Seção 2.3.3.3.

7.1.3 Complexidade Computacional

No algoritmo HubISC, a complexidade computacional pode ser representada como: $O(|X|^2 + (|Z|^2 + (|S| * |H|)))$. O cálculo da pontuação *hubness* para a fase offline é representado por $|X|^2$. O cálculo da pontuação *hubness* para a fase online é representado por $|Z|^2$. Observe que a pontuação *hubness* é calculada em *batches* na fase online, com base no *buffer* Z, e não é necessário calcular em todo o fluxo de dados de imagem S. Finalmente, $(|S| * |H|)$ refere-se à comparação de cada vetor $\vec{\boldsymbol{x_i}} \in S$ para os protótipos (*hubs*) de cada classe, onde $|H|$ é o número de hubs no modelo de decisão, $(|Z|^2 + (|S| * |H|))$ representa toda a complexidade da fase online.

7.2 Experimentos

Os cenários de experimentos propostos têm como objetivo avaliar o algoritmo HubISC em relação a outros algoritmos para a classificação de fluxos de dados de imagens.

Além disso, foram analisadas diferentes parametrizações do algoritmo HubISC. Para a realização dos experimentos foram considerados 7 conjuntos de imagens: *FASHION-MNIST (1), SVHN (2), CIFAR10 (3), LSUN (4), CINIC (5), COCO (6), TINY IMAGENET (7)* (veja Tabela 2).

Quanto à organização dos conjuntos de dados, o conjunto X, utilizado na fase *offline* dos algoritmos para a construção do modelo de decisão inicial, foi composto por 50% das classes do conjunto de imagens e 10% das imagens dessas respectivas classes. Assim, o fluxo de dados S, utilizado na fase *online*, é composto por 100% de classes presentes no conjunto de imagens e todas as imagens, com exceção das imagens selecionadas na fase *offline*, o que permite simular os desafios *concept-drift* e *concept-evolution*. As classes de imagens consideradas foram selecionadas aleatoriamente da mesma forma usada em (WANG et al., 2019).

O descritor CNN foi considerado para a extração de características das imagens. A arquitetura de rede selecionada para os experimentos foi *DenseNet*, com os mesmos parâmetros utilizados em (WANG et al., 2019). Para execução no algoritmo HubsISC foram considerados vetores de características com a mesma dimensionalidade de (WANG et al., 2019). O método definido no trabalho (LIMA et al., 2022) e descrito na Seção 2.4 foi utilizado para avaliação dos algoritmos. A eficácia dos algoritmos foi avaliada pelas métricas acurácia e *F-Measure* (veja Seção 2.4.3).

As diferentes análises experimentais estão descritas nas Seções 7.2.1, 7.2.2, 7.2.3 e 7.2.4. A Seção 7.2.1 realiza uma análise comparativa do algoritmo do HubISC em relação a outros algoritmos para classificação de fluxos de dados de imagens. A Seção 7.2.2 avalia no algoritmo HubISC as diferentes estratégias de esquecimento descritas na Seção 2.3.3.3. A Seção 7.2.3 analisa diferentes métodos de aprendizado ativo (veja a Seção 3.3) no Algoritmo HubISC. A Seção 7.2.4 analisa o comportamento do algoritmo HubISC em um conjunto de dados com ruído.

7.2.1 Avaliando o uso de *hubs* para a classificação de fluxos de dados de imagens

O experimento descrito aqui tem como objetivo investigar a seguinte questão de pesquisa: *Como o uso de hubs impacta no desempenho preditivo e na quantidade de instâncias a serem rotuladas quando comparado a outros algoritmos de classificação de fluxos de dados de imagens?* Para tanto, o algoritmo HubISC foi comparado com dois algoritmos comumente utilizados na classificação de fluxos de dados de imagens: NCM (versão original, sem otimizações) e CPE (veja Seção 3.1). Para o algoritmo HubISC, os parâmetros considerados foram: $L = 5$, $T = 1.000$, $K = 50$, h_K =70% da maior pontuação *hubness* obtida na fase *offline*. Os valores de L e T foram baseados em (WANG et al., 2019). As configurações dos algoritmos CPE e NCM foram definidas com base nos melhores

parâmetros informados pelos autores. Os resultados dos experimentos do primeiro cenário estão descritos na Tabela 16.

Tabela 16 – Acurácia (A), *F-Measure* (F) e Rótulos (R). Avaliação de diferentes algoritmos para classificação de fluxo de dados de imagens.

Conjunto de imagens	Algoritmos								
	CPE			NCM			HubISC		
	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)
1	**91,88**	**64,19**	20,63	82,43	59,44	15,31	89,12	63,23	**14,55**
2	**80,24**	**58,16**	22,62	72,84	51,63	16,89	78,86	55,45	**14,38**
3	**70,47**	**52,96**	27,15	67,72	48,51	18,79	69,63	51,39	**15,87**
4	**63,21**	**53,86**	22,13	56,71	48,41	21,98	63,15	50,87	**18,91**
5	**64,36**	**51,15**	23,62	51,72	45,27	20,35	61,54	48,19	**17,84**
6	**60,51**	**50,93**	31,24	50,19	45,58	28,96	58,43	48,17	**25,19**
7	**58,42**	**48,13**	29,22	52,78	44,91	27,68	57,34	45,88	**24,37**

Por meio da análise dos resultados apresentados na Tabela 16, verifica-se que o algoritmo CPE apresentou os maiores valores de *F-measure* e acurácia. Ao comparar os algoritmos CPE e NCM, a maior diferença encontrada foi na acurácia do conjunto de imagens CINIC (5), com uma diferença de 12,64%. Quando comparado ao algoritmo HubISC, a diferença na acurácia e na *F-measure* é menos evidente, a maior diferença foi encontrada na acurácia do conjunto de imagens LSUN (4) com um valor de 2,99%.

Apesar do algoritmo CPE ter apresentado os melhores resultados em termos de *F-measure* e acurácia, este algoritmo requer o maior número de rótulos informados, quando comparado aos algoritmos NCM e HubISC. O algoritmo HubISC solicitou o menor número de rótulos, com uma diferença de até 11,28% de rótulos no conjunto de imagens CIFAR10 (3). Foi observado que neste conjunto de imagens, a diferença na acurácia entre os algoritmos CPE e HubISC foi de apenas 0,84%. Essa diferença no número de rótulos informados é justificada pela utilização de *hubs* e pela sua influência na seleção das instâncias a serem rotuladas. Os resultados mostram que o algoritmo HubISC apresentou potencial considerável quando se trata de classificação de fluxo de dados de imagens, visto que apresenta acurácia e *F-measure* superiores em comparação ao algoritmo NCM, e menor acurácia e *F-measure* comparado ao algoritmo CPE. Porém, permite a utilização de um menor número de rótulos em comparação aos dois algoritmos. Para realizar uma análise mais profunda dos resultados do algoritmo HubISC, em termos de eficácia e rótulos, é essencial investigar a influência dos parâmetros h_K e L. Diferentes valores destes parâmetros foram avaliados nas Seções 7.2.1.1 e 7.2.1.2.

7.2.1.1 Avaliando diferentes valores do parâmetro h_K

O experimento descrito aqui tem como objetivo investigar a influência do parâmetro h_K nos resultados do algoritmo HubISC. A investigação deste parâmetro é interessante, pois impacta na quantidade de *hubs* que precisam ser identificados. Consequentemente, a

identificação dos *hubs* também influencia no número de rótulos a serem fornecidos (linhas 14 e 15 do Algoritmo 15).

No primeiro cenário de experimentos, descrito na Seção 7.2.1, o parâmetro h_K foi definido com base na maior pontuação *hubness* (h_P) identificada na fase *offline* do algoritmo HubISC da seguinte maneira: ($h_K = h_P * 0,7$). Instâncias de dados que as pontuações *hubness* $\geq h_K$ foram consideradas *hubs*. Este experimento avaliou diferentes valores de porcentagens para definição de h_K. A Tabela 17 mostra os resultados obtidos usando esta variação.

Como visto na Tabela 17, o parâmetro h_K influencia nos valores de acurácia e a *F-measure* do algoritmo HubISC. Além disso, esse parâmetro também impacta na quantidade de rótulos solicitados. Os maiores valores de acurácia e *F-measure* foram observados para 40% de h_K (Tabela 17). Porém, as maiores quantidades de rótulos também foram solicitadas nesta configuração. Para o cenário 90% de h_K (Tabela 17) foram encontrados os menores valores de acurácia, *F-measure* e rótulos.

O comportamento do parâmetro h_K identificado nos resultados se deve à sua influência na identificação de *hubs*. Quanto maior o valor de h_K, a identificação dos *hubs* se torna mais restritiva. Desta forma, o número de *hubs* é menor e a quantidade de rótulos solicitados também é menor. Assim, a tendência é que o algoritmo apresente valores menores de acurácia e *F-measure*. Por outro lado, quanto menor o valor de h_K, mais *hubs* são identificados e, consequentemente, mais rótulos são solicitados. Este fato ajuda a obter maiores valores de acurácia e *F-measure*.

Comparando os resultados do algoritmo CPE (Tabela 16), o algoritmo HubISC apresentou maiores valores de acurácia e *F-measure* em todos os conjuntos de imagens (exceto o conjunto de imagens CINIC (5), com uma diferença de 0,28%), para o caso de 40% de h_k. A diferença mais significativa observada foi de 2,98% na acurácia do conjunto de imagens LSUN (4). Porém, nesta configuração, o número de rótulos solicitados pelo algoritmo HubISC também foi maior em 4 conjuntos de dados (FASHION-MNIST(1), SVHN (2), CIFAR10 (3), TINY IMAGENET (7)), com a maior diferença no conjunto SVHN com 3,36%. No conjunto de imagens COCO (6), o valor de acurácia do algoritmo HubISC foi 1,55% maior com 3,06% menos rótulos solicitados. Este mesmo comportamento também foi observado no conjunto de imagens LSUN (4). Portanto, o uso do algoritmo baseado em *hubs* é promissor para classificação de fluxo de dados de imagens, considerando o potencial demonstrado em termos de desempenho preditivo e da quantidade de rótulos solicitada.

7.2.1.2 Avaliando diferentes valores do parâmetro L

O experimento descrito aqui tem como objetivo investigar a influência do parâmetro L nos resultados do algoritmo HubISC. Este parâmetro determina a distância máxima que uma instância pode estar de um *hub* para atribuição de rótulo. Caso ultrapasse

Tabela 17 – Acurácia (A), *F-Measure* (F) e Rótulos (R). Avaliação de diferentes valores do parâmetro h_K.

D	h_K																	
	40%			50%			60%			70%			80%			90%		
	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)
1	**92,51**	**64,77**	22,56	91,75	63,9	19,72	89,71	63,59	16,28	89,12	63,23	14,55	88,05	62,62	13,12	86,13	61,22	**10,98**
2	**81,78**	**58,32**	25,98	79,86	57,45	21,84	79,11	56,09	17,35	78,86	55,45	14,38	77,57	54,92	12,95	75,09	52,63	**11,12**
3	**71,66**	**53,23**	27,36	70,51	52,98	25,39	70,37	51,89	19,44	69,63	51,39	15,87	67,35	50,97	13,91	65,46	48,77	**11,98**
4	**66,19**	**53,45**	21,87	65,06	52,82	20,35	64,32	52,18	19,62	63,15	50,87	18,91	62,24	49,14	18,55	60,95	48,83	**17,12**
5	**64,08**	**51,38**	20,12	63,95	50,11	18,89	62,42	48,84	18,52	61,54	48,19	17,84	61,13	47,35	15,11	60,21	46,77	**14,18**
6	**62,06**	**50,16**	28,18	61,23	49,94	27,93	59,12	49,2	26,22	58,43	48,17	25,19	58,05	47,29	24,1	57,31	46,86	**23,41**
7	**61,15**	**49,14**	31,98	59,48	47,63	28,32	58,14	46,71	26,19	57,34	45,88	24,37	55,19	45,01	24,01	54,69	43,88	**23,12**

esta distância, a instância é adicionada ao *buffer* Z (linha 12 do Algoritmo 15), podendo também influenciar na identificação dos *hubs* e no número de rótulos solicitados. O parâmetro L foi definido como 5 no primeiro cenário de experimentos, conforme descrito na Seção 7.2.1, com base nos valores utilizados em (WANG et al., 2019). Este experimento avaliou diferentes valores para o parâmetro L. O valor de h_K foi mantido como 70%. A Tabela 18 mostra os resultados obtidos com esta variação.

Tabela 18 – Acurácia (A), *F-Measure* (F) e Rótulos (R). Avaliação de diferentes valores do parâmetro L.

D	L														
	1			3			5			7			9		
	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)
1	**94,11**	**67,96**	33,49	92,93	65,01	27,94	89,12	63,23	14,55	86,56	62,09	11,55	82,63	58,29	**8,43**
2	**85,63**	**62,43**	35,71	81,82	59,33	29,23	78,86	55,45	14,38	74,94	51,13	10,92	70,07	48,91	**7,66**
3	**73,48**	**56,39**	36,32	71,79	53,52	31,89	69,63	51,39	15,87	65,35	48,31	11,79	60,86	45,12	**8,2**
4	**69,18**	**60,22**	30,88	66,61	54,71	25,15	63,15	50,87	18,91	60,48	47,14	15,01	57,7	45,86	**10,34**
5	**67,38**	**53,94**	34,19	65,24	51,17	29,1	61,54	48,19	17,84	60,01	46,41	12,66	57,3	44,03	**7,15**
6	**65,03**	**53,81**	42,19	60,75	51,22	34,87	58,43	48,17	25,19	55,49	44,63	20,66	51,17	39,1	**12,58**
7	**65,71**	**50,08**	38,97	61,18	48,21	30,28	57,34	45,88	24,37	54,78	42,69	18,68	51,6	41,15	**14,12**

Por meio da análise dos resultados apresentados na Tabela 18, nota-se que o parâmetro L impacta no número de rótulos solicitados pelo algoritmo HubISC. Além disso, influencia nos valares de acurácia e *F-measure*. Os maiores valores de acurácia e *F-measure* foram encontrados para $L = 1$. Com esta configuração também foi detectado o maior número de rótulos solicitados. No cenário com valor $L = 9$, foram identificados o menor número de rótulos e os menores valores de acurácia e *F-measure*.

A influência do parâmetro L se deve ao seu impacto na seleção de instâncias a serem armazenadas no *buffer* Z. Vale ressaltar que quanto menor o valor de L, maior será o número de instâncias inseridas em Z. Portanto, a tendência é que Z seja preenchido com mais frequência no processamento do fluxo de dados de imagens. Assim, mais rótulos são solicitados e o algoritmo pode apresentar maiores valores de acurácia e *F-measure*. Por outro lado, quanto maior o valor de L, menos instâncias são inseridas em Z. Assim, menos *hubs* são identificados, menos rótulos são solicitados e o algoritmo tende a apresentar menor eficácia com esta configuração.

Comparando os resultados apresentados na Tabela 18 com o algoritmo CPE (Tabela 16), nas configurações $L = 1$ e $L = 3$, o algoritmo HubISC apresentou valores de acurácia e *F -measure* maiores em todos os conjuntos de imagens. Foi observada uma melhora de até 7,29% na acurácia do conjunto de imagens TINY IMAGENET (7). Porém, nesta configuração, o número de rótulos solicitados pelo algoritmo HubISC foi até 9,75% maior.

7.2.2 Avaliando diferentes métodos de esquecimento de representantes

Nos experimentos anteriores, o algoritmo HubISC não considerou métodos de esquecimento de representantes. Porém, em ambientes de fluxo de dados, é importante utilizar este recurso para atualizar o modelo de decisão (veja Seção 2.3.3.3). Assim, esta seção de experimentos tem como objetivo explorar a seguinte questão de pesquisa: *"o uso de métodos de esquecimento de representantes pode influenciar no desempenho preditivo e na quantidade de rótulos solicitados pelo algoritmo HubISC?"* Os métodos de esquecimento considerados foram os mesmos descritos na Seção 2.3.3.3. Neste cenário de experimentos, o valor $L = 5$ foi selecionado por representar um caso tipicamente médio do algoritmo HubISC, com valores interessantes para número de rótulos e acurácia. Além disso, h_K foi calculado considerando 70%. A Tabela 19 descreve os resultados dos experimentos que exploraram as estratégias de esquecimento.

Tabela 19 – Acurácia (A), *F-Measure* (F) e Rótulos (R). Avaliação de diferentes estratégias de esquecimento de representantes: (1) Todos os representantes; (2) Proximidade; (3) Últimas requisições; (4) Pontuação *Hubness*; (5) Últimas requisições com memória de esquecimento.

D	Estratégia														
	(1)			(2)			(3)			(4)			(5)		
	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)
1	89,1	63,2	**14,6**	90,5	63,9	18,9	91,5	64,6	18,2	90,4	64,2	18,3	**91,9**	**64,9**	17,1
2	78,9	55,5	**14,4**	79,6	56,8	20,9	80,2	58	20,8	79,9	57,8	20,8	**80,6**	**58,3**	18,9
3	69,6	51,4	**15,9**	70	52,1	20	70,4	52,9	19,1	70,2	52,4	19,3	**71,1**	**52,9**	18,5
4	63,15	50,87	**18,91**	64,22	51,42	20,45	64,05	52,17	20,13	63,75	52,05	21,09	**64,57**	**52,39**	19,76
5	61,54	48,19	**17,84**	62,31	49,12	19,42	61,13	48,98	19,35	63,02	49,55	21,96	**63,92**	**50,6**	19,4
6	58,43	48,17	**25,19**	60,1	48,7	28,74	60,29	49,58	29,03	59,98	48,63	28,18	**61,19**	**50,09**	27,14
7	57,34	45,88	**24,37**	59,91	47,13	24,43	58,17	46,15	24,65	59,35	47,2	25,19	**60,28**	**49,83**	24,44

O primeiro resultado da Tabela 19 refere-se à estratégia de todos os protótipos (representantes). Nenhum mecanismo de esquecimento foi considerado. Este resultado é o mesmo da Tabela 18 para o parâmetro $L = 5$. Por meio da análise dos resultados das estratégias de esquecimento, nota-se que o número de rótulos solicitados aumentou em todos os métodos, quando comparado ao caso base (todos os protótipos). Este fato ocorre pois pode ser necessário solicitar o rótulo de novos protótipos, o que não aconteceria se todos os protótipos fossem mantidos. Contudo, é importante ressaltar que manter todos os protótipos na memória pode ser impraticável, considerando as restrições associadas aos algoritmos de fluxo de dados. Apesar do aumento do número de rótulos solicitados, também foi observado um aumento na acurácia e nos valores da *F-measure* em todos os conjuntos de dados e em todas as estratégias de esquecimento em relação ao caso base. Os melhores resultados de acurácia e *F-measure* foram observados ao usar a estratégia que considera as últimas requisições com memória de esquecimento em todos os conjuntos de dados. Por exemplo, no conjunto de imagens TINY IMAGENET (7), foi observado um

aumento de 2,94% na acurácia e de 0,07% nos rótulos solicitados. Com isso, percebe-se a importância e a influência da implementação da memória de esquecimento nesses métodos. A capacidade da memória de esquecimento foi definida como $T/2$ nos experimentos.

Nos experimentos das seções anteriores, o algoritmo HubISC, apesar de solicitar menos rótulos, não superou o algoritmo CPE em termos de acurácia e *F-measure* em todos os conjuntos de dados. Portanto, para fazer uma comparação direta entre o algoritmo CPE e os resultados da última versão do algoritmo HubISC, é necessário comparar os melhores resultados do algoritmo HubISC, considerando a incorporação de estratégias de esquecimento (três últimas colunas da Tabela 19), com os resultados do algoritmo CPE (Tabela 16). A partir dos resultados desta comparação, o número de rótulos solicitados pelo algoritmo HubISC permaneceu menor do que quando comparado ao algoritmo CPE. Em todos os conjuntos de dados, os valores de acurácia e *F-measure* do algoritmo HubISC foram superiores, o que reforça a importância da implementação de métodos de esquecimento em algoritmos de fluxo de dados.

A Figura 20 compara os resultados dos algoritmos HubISC e CPE. Este gráfico compara a quantidade de rótulos solicitados e a acurácia apresentada pelos algoritmos no conjunto de imagens CIFAR10. Pelo gráfico, nota-se que o número de rótulos solicitados pelo algoritmo HubISC foi menor ao longo do fluxo. Em relação à acurácia, observa-se um equilíbrio, mas o algoritmo HubISC se destacou principalmente na metade inicial do fluxo, o que indica a capacidade do algoritmo HubISC em lidar com o surgimento de novas classes.

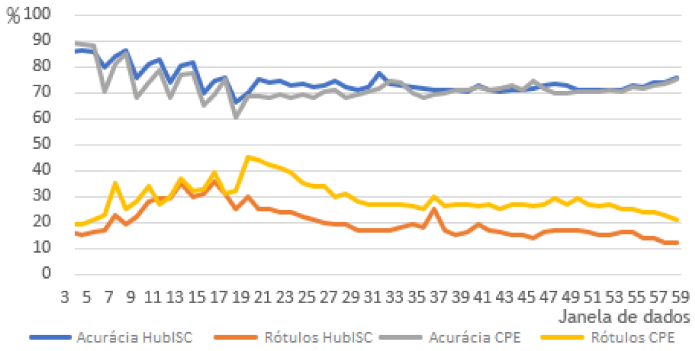

Figura 20 – CIFAR10: Comparando HubISC x CPE.

A Figura 21 ilustra a organização do conjunto CIFAR10 e como as instâncias de dados foram apresentadas aos algoritmos. Nesta figura, cada cor representa uma classe de imagem. É importante ressaltar que algumas classes são apresentadas inicialmente com um conjunto de instâncias (fase *offline*). Outras classes são apresentadas apenas no fluxo de dados (fase *online*). A última classe do conjunto de dados foi apresentada no tempo 20 (20.000). Comparando este tempo com a Figura 20, as oscilações mais significativas nos valores de acurácia e *F-measure* ocorrem até este ponto e depois uma estabilidade é observada. A partir da análise da Figura 21 observa-se a capacidade do algoritmo HubISC

de lidar com o fenômeno *concept-evolution*.

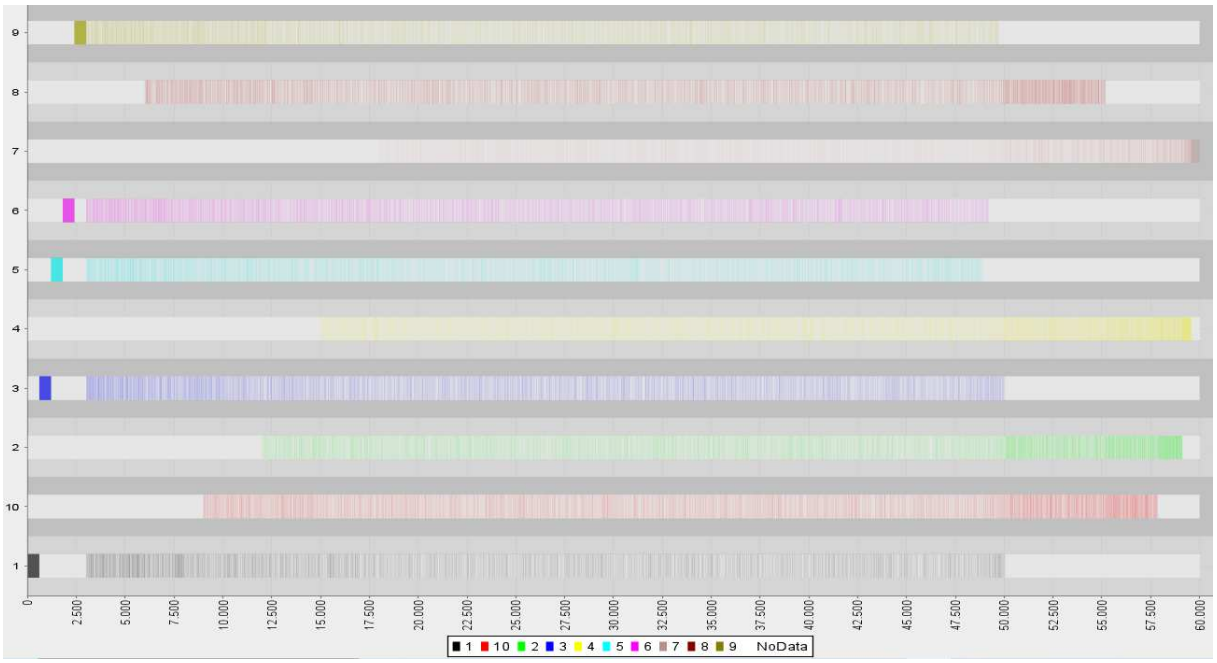

Figura 21 – CIFAR10: Organização do conjunto de dados.

7.2.3 Avaliando diferentes estratégias de aprendizado ativo

Nos experimentos anteriores, o algoritmo HubISC considerou a pontuação *hubness* das instâncias de dados para a seleção de instâncias a serem rotuladas como uma estratégia de aprendizado ativo. Entretanto, outras estratégias de aprendizado ativo não foram analisadas. Dessa forma, comparar a estratégia baseada em *hubs* com outras estratégias de aprendizado ativo, comumente utilizadas na literatura científica da área, torna-se um fator interessante. Os experimentos descritos aqui visam explorar a seguinte questão de pesquisa: *Como o uso de hubs impacta no desempenho preditivo e na escolha das instâncias a serem rotuladas quando comparado ao uso de outras estratégias de aprendizado ativo?* As estratégias de aprendizado ativo usadas para comparação foram as mesmas descritas na Seção 3.3. É importante ressaltar que esses experimentos já consideram a implementação do método de esquecimento de representantes implementado na seção anterior.

As estratégias (1) e (2), apresentadas na Tabela 20, consideram a seleção aleatória de instâncias (veja Algoritmo 10). Essa estratégia é simples, pois as instâncias são selecionadas aleatoriamente sem analisar o grau de informatividade da instância dos dados. Dois cenários foram considerados para os experimentos: (1) sem *budget* e (2) com *budget=20*. O valor de *budget* visa definir um limite de instâncias que a estratégia pode selecionar. Por meio da análise da Tabela 20, os resultados da estratégia sem *budget* foram superiores em termos de acurácia e *F-measure*, mas o número de rótulos solicitados foi próximo de 50%, o que é impraticável em qualquer situação real. O número de rótulos solicitados com *budget=20* foi semelhante ao da estratégia *hubness*, mas a acurácia e os

valores de *F-measure* foram inferiores em todos os conjuntos de dados. Este comportamento já era esperado devido à simplicidade dessa estratégia de aprendizado ativo.

Tabela 20 – Acurácia (A), *F-Measure* (F) e Rótulos (R). Avaliação de diferentes estratégias de aprendizado ativo: (1) Aleatória; (2) Aleatória - *budget=20*; (3) Incerteza - *threshold=0,5*; (4) Incerteza - *threshold=0,7*; (5) Custo.

D	Estratégia														
	(1)			(2)			(3)			(4)			(5)		
	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)	(A)	(F)	(R)
1	**95,71**	**70,1**	51,3	90,18	63,89	20	93,6	68,12	30,1	91,72	63,93	21,13	89,17	61,12	**14,85**
2	**85,66**	**69,1**	50,6	77,63	54,77	20	82,17	61,34	33,6	79,92	57,82	23,29	79,19	57,05	**15,89**
3	**79,92**	**60,16**	50,9	68,81	49,08	20	72,56	53,87	32,1	70,32	52,28	25,98	70,33	50,58	**16,76**
4	**78,19**	**61,14**	49,6	61,18	48,02	20	73,29	56,32	36,3	65,49	54,11	27,71	65,29	52,94	**17,31**
5	**75,47**	**59,33**	51,1	59,52	50,37	20	71,13	53,19	37,1	62,39	52,49	23,46	61,05	51,78	**18,33**
6	**70,4**	**58,88**	50,3	55,11	47,13	**20**	65,13	55,12	42,18	63,45	54,17	34,95	59,17	48,97	26,48
7	**65,41**	**57,12**	49,8	56,38	45,99	**20**	62,29	54,93	39,09	61,09	50,88	30,02	60,09	49,11	22,17

As estratégias (3) e (4), apresentadas na Tabela 20, considera o grau de incerteza da classificação (veja Algoritmo 11). Nesta estratégia é necessário informar um limiar que determine o grau de incerteza a ser considerado na seleção das instâncias de dados. Um menor grau de incerteza é menos restritivo, ou seja, mais instâncias de dados serão selecionadas. Um maior grau de incerteza selecionará menos instâncias de dados. Nos experimentos foram considerados valores de 0,5 (Estratégia 3) e 0,7 (Estratégia 4) como limite. A partir dos resultados da Tabela 15, o valor de 0,5 foi apontado como menos restritivo e que mais instâncias de dados foram selecionadas, o que aumentou a acurácia e os valores de *F-measure*. No entanto, a quantidade de rótulos solicitados pode ser impraticável em cenários reais de aplicações. Com o limite de 0,7, menos instâncias foram selecionadas, mas a acurácia e os valores de *F-measure* foram inferiores à estratégia *hubness* em todos os conjuntos de dados.

A última estratégia explorada de aprendizado ativo baseou-se no custo (veja Algoritmo 11) de obtenção do rótulo (Estratégia (5) da Tabela 20). De acordo com os resultados de (PARREIRA; PRATI, 2019) esta é uma das estratégias mais promissoras. Porém, a forma de calcular o custo associado à obtenção do rótulo das instâncias de dados pode variar de acordo com as características do algoritmo de classificação. No caso do algoritmo HubISC, o cálculo do custo foi considerado da seguinte forma: calcula-se a pontuação *hubness* e observa-se o rótulo das instâncias vizinhas. Assim, o custo para obtenção do rótulo é proporcional ao conceito de *bad-hubness* (veja Seção 2.2). Por meio da análise dos resultados da estratégia baseada em custos, e comparando-os com os resultados originais do HubISC (Tabela 19), nota-se que a estratégia com custo apresentou um menor número de rótulos solicitado e também valores inferiores de acurácia e *F-measure*. No entanto, essa estratégia pode oferecer uma relação custo-benefício interessante. Por exemplo, no conjunto de imagens TINY IMAGENET (7) o número de rótulos caiu 2,27% e o valor de acurácia diminuiu apenas 0,19%, o que pode ser interessante, dependendo da disponibilidade do usuário especialista de fornecer as informações de rótulo.

Por meio da análise das diferentes estratégias de aprendizado ativo, aplicadas no algoritmo HubISC, observou-se que a estratégia que considera os *hubs* como instâncias a serem rotuladas (três últimas colunas da Tabela 19) é capaz de produzir resultados interessantes em termos de desempenho preditivo e também de quantidade de rótulos solicitados. Além disso, destaca-se que a identificação de *hubs* é um recurso natural do algoritmo HubISC, o que evita a necessidade de adicionar recursos de outras estratégias de aprendizado ativo.

7.2.4 Avaliando conjunto de imagens com ruído

Conforme descrito na Seção 2.2, algoritmos que consideram *hubs* podem ser capazes de trabalhar em conjuntos de dados ruidosos devido à identificação de instâncias de dados distantes das demais. Assim, os experimentos aqui descritos visam investigar a seguinte questão de pesquisa: "*O algoritmo HubISC é capaz de apresentar uma eficácia superior ao algoritmo CPE em um conjunto de imagens com ruído?*". Os experimentos descritos aqui empregaram a versão modificada do conjunto de imagens CIFAR10 que possui diferentes porcentagens de ruído (JIANG et al., 2020). A configuração do algoritmo HubISC considerou a estratégia de aprendizado ativo baseada em *hubs* e o mecanismo de esquecimento que apresentou os melhores resultados nos experimentos anteriores (últimas requisições com memória de esquecimento). A Tabela 21 descreve os resultados obtidos nesses experimentos.

Tabela 21 – Acurácia (A), *F-Measure* (F) e Rótulos (R). Comparação HubISC x CPE no conjunto CIFAR10 com ruídos.

Nível de ruído	HubISC			CPE		
	(A)	(F)	(R)	(A)	(F)	(R)
0%	**71,05**	52,88	**18,48**	70,47	**52,96**	27,15
1%	**69,84**	**52,05**	**20,94**	69,05	51,58	28,62
5%	**68,01**	**49,98**	**23,75**	67,91	48,61	30,29
10%	**63,02**	**45,05**	**26,41**	62,15	44,09	34,18
20%	**60,98**	**42,31**	**32,30**	59,49	42,23	39,93

Os resultados apresentados na Tabela 21 mostram que os algoritmos HubISC e CPE apresentaram degradações graduais a cada nível de ruído introduzido (1%, 5% 10% e 20%). Além disso, o número de rótulos solicitados também aumentou. Ao comparar os dois algoritmos, nota-se que eles sofreram degradações semelhantes, por exemplo, 10,07% na acurácia do algoritmo HubISC e 10,98% no algoritmo CPE com 20% de ruído (comparando os níveis de ruído 0% e 20%). Destaca-se que algoritmo HubISC ainda apresentou menor dependência de rótulo quando comparado ao algoritmo CPE em todos os níveis de ruído. De forma a complementar a análise dos resultados, foi realizado o teste estatístico descrito na Seção 6.2.2.5.

7.2.5 Análise estatística

A análise estatística realizada na Seção 7.2.5.1 comparou os resultados dos diferentes cenários de experimentos realizados, considerando os valores de acurácia e *F-measure*. Apesar da importância dessa análise, uma das características do algoritmo HubISC é a menor dependência de rótulos em relação aos algoritmos comparados. Dessa forma, na Seção 7.2.5.2 foi realizada uma análise estatística complementar, que além de considerar o desempenho preditivo, utiliza o número de rótulos solicitados pelos algoritmos para a análise estatística. Nas duas análises foi aplicado o *Wilcoxon signed-rank test* (veja Seção 4.3.2).

7.2.5.1 Avaliando desempenho preditivo

A hipótese nula formulada para esta análise estatística foi: H_0 − os desempenhos preditivos apresentados em dois cenários de experimentos (A e B) não são estatisticamente diferentes, a um nível de confiança de 95%. A hipótese alternativa formulada foi: H_1 − o desempenho preditivo apresentado no cenário de experimentos A é superior ao desempenho preditivo apresentado no cenário de experimentos B.

A primeira análise realizada comparou os resultados obtidos pelos diferentes algoritmos de classificação de fluxos de dados de imagens (Tabela 16). Os resultados foram comparados da seguinte forma: CPE x NCM, HubISC x NCM e HubISC x CPE. Em todos os casos, os resultados foram estatisticamente diferentes, ou seja, H0 foi rejeitada. Isso confirma a superioridade do algoritmo CPE em relação aos demais algoritmos neste cenário de experimentos e também a superioridade do algoritmo HubISC em relação ao NCM.

A segunda análise estatística comparou os resultados do algoritmo HubISC em sua versão original (Tabela 16) com a versão do algoritmo HubISC com a estratégia de esquecimento de representantes (Tabela 19). Por meio dessa análise, observou-se a superioridade da versão do algoritmo HubISC que considera a estratégia de últimas requisições com memória de esquecimento.

A terceira análise estatística comparou os resultados do algoritmo CPE com os resultados do algoritmo HubISC com a estratégia de esquecimento de representantes. Estes resultados estão na Tabela 16 (resultados CPE) e na Tabela 19 (últimas três colunas). Foi observada diferença estatística entre os resultados dos algoritmos, o que indica que os resultados do algoritmo HubISC superaram o algoritmo CPE a partir da implementação da estratégia de esquecimento.

Na quarta análise estatística foram analisadas as diferentes estratégias de aprendizado ativo implementadas no algoritmo HubISC (Tabela 20). A versão original do algoritmo HubISC foi utilizada para a comparação com as demais estratégias. Em todos os casos foi observada diferença estatística entre os resultados. A estratégia original, que

considera *hubs*, se mostrou superior as demais estratégias, com exceção da estratégia Aleatória (sem *budget*).

Finalmente, a última análise estatística comparou os resultados dos algoritmos HubISC e CPE no conjunto de imagens CIFAR10 com ruído (Tabela 21). Também foi observada nesta análise diferença estatística entre os resultados obtidos pelos dois algoritmos, o que permitiu verificar a superioridade do algoritmo HubISC em termos de desempenho preditivo.

7.2.5.2 Avaliando desempenho preditivo e número de rótulos

Para a análise estatística descrita aqui foi necessário utilizar uma métrica para combinar o desempenho preditivo e o número de rótulos solicitados pelos algoritmos. Para tanto, foi utilizada a métrica ν definida em (PULLAR-STRECKER et al., 2022), que está representada na Equação 11:

$$\nu(\alpha, \iota, \varrho, \zeta, \eta) = (1 - \alpha) \cdot \zeta \cdot \eta + \iota \cdot \varrho \tag{11}$$

Na Equação 11, α representa o desempenho preditivo dos algoritmos, que aqui pode-se considerar acurácia ou *F-Measure*, ι representa o número de rótulos solicitados pelos algoritmos, ϱ representa o custo para a obtenção de um único rótulo, ζ representa o custo por predição incorreta do algoritmo de classificação. Por fim, η representa o número total de predições realizadas pelo algoritmo de classificação. Ao final do cálculo, os valores de ν podem ser normalizados no intervalo [0,1], por exemplo, por meio das técnicas *Zscore* ou *MinMax*. Destaca-se por meio da Equação 11, que o objetivo é minimizar o valor de ν, ou seja, quanto menor o valor de ν, melhor é o desempenho do algoritmo analisado. É importante ressaltar que os valores de ϱ e ζ foram definidos como 1 na análise realizada, pois os custos não foram o foco da investigação nos algoritmos NCM, CPE e HubISC.

Considerando o objetivo dessa análise estatística, a hipótese nula formulada foi: H_0 − considerando os valores da métrica ν, os desempenhos apresentados em dois cenários de experimentos (A e B) não são estatisticamente diferentes, a um nível de confiança de 95%. A hipótese alternativa formulada foi: H_1 − considerando os valores da métrica ν, o desempenho apresentado no cenário de experimentos A é superior ao desempenho apresentado no cenário de experimentos B.

A Tabela 22 apresenta os valores da métrica ν calculados com base nos valores apresentados na Tabela 16, que comparou os resultados de diferentes algoritmos para a classificação de fluxos de dados de imagens. É importante ressaltar que os valores da métrica ν foram calculados separadamente para acurácia e *F-measure*. Além disso, os valores de ν foram normalizados, por meio do método *MinMax*, na escala [0,1] com 3 casas decimais. Por meio da análise estatística realizada, utilizando os valores apresentados na Tabela 22, a hipótese nula (H_0) foi rejeitada em todas as comparações. Dessa forma,

notou-se a superioridade do algoritmo CPE em relação aos algoritmos HubISC e NCM. Além disso, observou-se a superioridade do algoritmo HubISC em relação ao NCM.

Tabela 22 – $\nu(A)$ - Acurácia e $\nu(F)$ - *F-Measure*. Avaliação de diferentes algoritmos para classificação de fluxo de dados de imagens. Valores calculados com base na Tabela 16.

Conjunto de imagens	Algoritmos					
	CPE		NCM		HubISC	
	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$
1	**0,000**	**0,000**	0,047	0,026	0,014	0,005
2	**0,169**	**0,201**	0,220	0,250	0,179	0,221
3	**0,091**	**0,024**	0,102	0,043	0,095	0,031
4	**0,284**	**0,230**	0,329	0,272	0,285	0,253
5	**0,285**	**0,258**	0,371	0,302	0,304	0,280
6	**0,873**	**0,929**	1,000	1,000	0,899	0,965
7	**0,354**	**0,307**	0,390	0,329	0,361	0,322

Na Tabela 23 são apresentados os valores da métrica ν que foram calculados com base nos valores apresentados na Tabela 19. O objetivo dessa análise foi comparar o desempenho das diferentes estratégias de esquecimento de representantes implementadas no algoritmo HubISC. Considerando a análise estatística realizada, a hipótese nula (H_0) foi rejeitada em todas as comparações utilizando a Estratégia (5) como base, sendo possível observar a superioridade da Estratégia (5) (Últimas requisições com memória de esquecimento) em relação as demais estratégias comparadas.

Tabela 23 – $\nu(A)$ - Acurácia e $\nu(F)$ - *F-Measure*. Avaliação de diferentes estratégias de esquecimento de representantes: (1) Todos os representantes; (2) Proximidade; (3) Últimas requisições; (4) Pontuação *Hubness*; (5) Últimas requisições com memória de esquecimento. Valores calculados com base na Tabela 19.

D	Estratégia									
	(1)		(2)		(3)		(4)		(5)	
	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$
1	0,017	0,011	0,009	0,006	0,002	0,002	0,009	0,004	**0,000**	**0,000**
2	0,178	0,220	0,172	0,209	0,167	0,198	0,170	0,200	**0,164**	**0,195**
3	0,101	0,030	0,099	0,026	0,097	0,022	0,098	0,024	**0,093**	**0,022**
4	0,346	0,285	0,337	0,280	0,338	0,274	0,341	0,275	**0,334**	**0,272**
5	0,353	0,302	0,346	0,294	0,356	0,295	0,340	0,290	**0,332**	**0,281**
6	1,000	1,000	0,974	0,992	0,971	0,978	0,976	0,993	**0,958**	**0,970**
7	0,426	0,353	0,406	0,343	0,420	0,350	0,411	0,342	**0,404**	**0,322**

Após a análise das estratégias de esquecimento de representantes, pode-se realizar uma comparação direta entre os resultados dos algoritmos HubISC e CPE, considerando o novo recurso implementado no algoritmo HubISC. Para tanto, a Tabela 24 apresenta

os valores da métrica ν para dois algoritmos. Os valores de ν foram calculados com base na Tabela 19 para o algoritmo HubISC e com base na Tabela 16 para o algoritmo CPE. Por meio da análise estatística realizada, observou-se que a hipótese nula (H_0) foi rejeitada. Dessa forma, notou-se que após a implementação da estratégia de esquecimento de representantes, os resultados do algoritmo HubISC apresentaram superioridade em relação ao algoritmo CPE.

Tabela 24 – $\nu(A)$ - Acurácia e $\nu(F)$ - *F-Measure*. Comparação HubISC x CPE. Valores do algoritmo HubISC calculados com base na Tabela 19. Valores do algoritmo CPE calculados com base na Tabela 16.

Conjunto de imagens	CPE		HubISC	
	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$
1	0,038	0,032	**0,000**	**0,000**
2	0,225	0,241	**0,157**	**0,191**
3	0,139	0,057	**0,089**	**0,021**
4	0,351	0,272	**0,320**	**0,266**
5	0,352	0,301	**0,319**	**0,276**
6	1,000	1,000	**0,918**	**0,950**
7	0,428	0,352	**0,387**	**0,315**

Tabela 25 – $\nu(A)$ - Acurácia e $\nu(F)$ - *F-Measure*. Avaliação de diferentes estratégias de aprendizado ativo: (1) Aleatória; (2) Aleatória - *budget=20*; (3) Incerteza - *threshold=0,5*; (4) Incerteza - *threshold=0,7*; (5) Custo; (6) *Hubs*. Valores calculados com base na Tabela 20. Os valores da estratégia (6) foram calculados com base na Tabela 19.

D	Estratégia											
	(1)		(2)		(3)		(4)		(5)		(6)	
	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$
1	0,171	0,122	0,186	0,140	0,177	0,128	0,182	0,139	0,189	0,147	**0,000**	**0,000**
2	0,327	0,284	0,358	0,342	0,341	0,316	0,349	0,330	0,352	0,333	**0,122**	**0,155**
3	0,164	0,091	0,190	0,118	0,181	0,106	0,186	0,110	0,186	0,114	**0,070**	**0,017**
4	0,350	0,311	0,416	0,366	0,369	0,331	0,399	0,341	0,400	0,345	**0,250**	**0,216**
5	0,369	0,326	0,429	0,362	0,385	0,351	0,418	0,354	0,423	0,357	**0,249**	**0,223**
6	0,883	0,904	1,000	1,000	0,923	0,935	0,936	0,943	0,969	0,985	**0,717**	**0,770**
7	0,400	0,329	0,435	0,375	0,412	0,338	0,417	0,355	0,421	0,362	**0,302**	**0,255**

A Tabela 25 apresenta os valores da métrica ν para a comparação entre diferentes estratégias de aprendizado ativo implementadas no algoritmo HubISC. Tal análise foi detalhada na Tabela 20. Para a estratégia de aprendizado ativo, originalmente utilizada no algoritmo HubISC, que considera o uso de *hubs* (Estratégia (6) da Tabela 25), foram considerados os resultados das colunas identificadas como (5) na Tabela 19. Por meio da análise estatística realizada, a hipótese nula (H_0) foi rejeitada em todas as comparações

utilizando a Estratégia (6) como base. Esse fato indica que a estratégia que considera o uso de *hubs* para o aprendizado ativo apresentou desempenho superior em relação as demais estratégias que foram utilizadas no algoritmo HubISC.

A última análise estatística realizada considerou os valores de ν apresentados na Tabela 26. Nessa análise foi realizada a comparação entre os desempenhos dos algoritmos HubISC e CPE no conjunto CIFAR10 com ruídos. O cálculo da métrica ν foi realizada com base nos valores apresentados na 21. Na análise estatística realizada, a hipótese nula (H_0) foi rejeitada. Então, observou-se a superioridade dos resultadso do algoritmo HubISC em relação ao algoritmo CPE em um conjunto de dados com ruídos.

Tabela 26 – $\nu(A)$ - Acurácia e $\nu(F)$ - *F-Measure*. Comparação HubISC x CPE no conjunto CIFAR10 com ruídos. Valores calculados com base na Tabela 21.

Nível de ruído	CPE		HubISC	
	$\nu(A)$	$\nu(F)$	$\nu(A)$	$\nu(F)$
0%	0,026	0,000	**0,000**	**0,004**
1%	0,154	0,133	0,119	0,108
5%	0,276	0,355	0,271	0,283
10%	0,677	0,724	0,642	0,676
20%	1,000	1,000	0,944	0,996

7.3 Considerações Finais

A classificação de fluxos de dados de imagens é uma tarefa importante que ainda possui questões em aberto para serem exploradas. Nesse contexto, uma dessas questões é a alta dimensionalidade dos dados. A maior parte dos estudos da literatura científica da área não aborda esse desafio. O algoritmo HubISC, desenvolvido neste trabalho de doutorado, para a classificação de fluxos de dados de imagem, incorpora o aspecto *hubness*, inerente de dados de alta dimensão. O algoritmo HubISC apresenta características interessantes para a classificação de fluxos de dados de imagens:

- possui uma estrutura de sumarização das instâncias de dados das classes por meio da utilização dos *hubs*;
- apresenta um recurso natural para a seleção de instâncias no processo de aprendizado ativo, pois os *hubs* podem representar essas instâncias de dados;
- apresenta desempenho preditivo superior quando comparado a outros algoritmos de classificação de fluxo de dados de imagens;
- permite o tratamento dos fenômenos *concept-evolution* e *concept-drift* a partir da atualização online do modelo de decisão;
- possui parâmetros (h_K e L) que podem ajustar a quantidade de rótulos solicitados, o que pode contribuir para cenários de aplicações reais com disponibilidade limitada

de usuários especialistas;
- pode ser parametrizado com diferentes estratégias de aprendizado ativo;
- pode ser estendido com diferentes métodos de esquecimento de representantes.

Por meio da análise dos resultados experimentais obtidos com o algoritmo HubISC e da análise estatística realizada, nota-se que o algoritmo HubISC apresentou resultados superiores em termos de desempenho preditivo. Além disso, destaca-se que o algoritmo HubISC demandou uma menor quantidade de rótulos em comparação aos algoritmos CPE e NCM.

Capítulo 8

Conclusão

Atualmente, com a alta disponibilidade de dispositivos de aquisição de imagens, surgiu a necessidade do desenvolvimento de estratégias para lidar com fluxos de dados de imagens. O trabalho descrito nesta tese teve como principal objetivo contribuir para a classificação de fluxos de dados de imagens nas etapas de classificação, atualização do modelo e avaliação considerando aspectos inerentes de cenários de aplicações reais.

A condução do trabalho foi realizada em três etapas. A primeira etapa abordou a hipótese de pesquisa **H1**, apresentada no Capítulo 1: *o desempenho preditivo de algoritmos de classificação de fluxos de dados de imagens sofre a influência da quantidade de imagens e de classes de imagens utilizada na construção dos descritores de características.* Esta etapa foi explicada no Capítulo 5, que apresentou um estudo experimental com os descritores de características BoVW e CNN para a representação de imagens na classificação de fluxos de dados imagens. Em ambientes de fluxos de dados existem ainda mais desafios, pois é necessário lidar com os fenômenos *concept-drift* e *concept-evolution*. Por exemplo, considerando os descritores BoVW (veja Seção 2.1.2) e CNN (veja Seção 2.1.3), a amostra de dados considerada para a construção desses descritores pode não ser suficientemente representativa para todas as imagens de um fluxo. Dessa forma, é importante analisar a influência das imagens e classes de imagens consideradas para a construção desses descritores. Os resultados obtidos com esse estudo permitiram constatar que a quantidade de imagens e de classes de imagens utilizadas na construção dos descritores possui influência na qualidade da descrição de imagens e, consequentemente, no resultado da classificação. Os piores resultados constatados nesse estudo experimental, em termos de eficácia do classificador, foram observados com as menores quantidades de classes e também de imagens utilizadas para a construção dos descritores de características. Outro fato observado é que um algoritmo que considera a evolução de características (*feature-evolution*) pode amenizar a degradação da eficácia. Além disso, notou-se por meio dos resultados, que com o descritor de características CNN os classificadores apresentaram maior eficácia.

A segunda etapa do trabalho explorou a hipótese de pesquisa **H2**, apresentada no Capítulo 1: *os algoritmos de fluxos de dados de imagens da literatura científica da*

área, quando submetidos a cenários de aplicação reais, têm o seu desempenho preditivo degradado. Esta etapa foi descrita no Capítulo 6. Durante a revisão da literatura científica da área para a classificação de fluxos de dados de imagens, constatou-se que os métodos utilizados para a avaliação de algoritmos de classificação de fluxos de dados de imagens não são adequados para cenários de aplicações reais. Com o intuito de contribuir com uma solução para essa questão, foi desenvolvido um *framework* de avaliação apropriado para ambientes em que não se conhece todas as classes de imagens (*concept-evolution*), não se tem o rótulo de todas as imagens para a atualização do modelo e os conceitos podem mudar ao longo do tempo (*concept-drift*). Como resultado foi criado o *framework* EVISClass, que considera tais cenários para a avaliação de classificadores de fluxos de imagens. Durante os experimentos foram considerados diferentes algoritmos de classificação de fluxos de dados de imagens. Com os resultados experimentais obtidos foi possível constatar que quanto maior a realidade considerada nos métodos de avaliação, maior a degradação da eficácia dos classificadores avaliados. Além disso, este estudo complementou a análise realizada no Capítulo 5, utilizando o *framework* EVISClass para a avaliar o refinamento do descritor de características CNN. Por meio dos resultados experimentais nesse estudo complementar percebeu-se que as maiores degradações dos resultados foram observadas nos casos com os maiores valores de latência e as menores quantidades de imagens para o refinamento do descritor de características.

A terceira etapa do trabalho abordou a hipótese de pesquisa **H3**, apresentada no Capítulo 1: *o uso de hubs como estrutura de sumarização de dados e como instâncias a serem rotuladas na etapa de aprendizado ativo leva a uma diminuição do número de instâncias rotuladas e em um melhor desempenho preditivo na classificação de fluxos de dados de imagens.* Essa etapa foi abordada no Capítulo 7, que descreveu a criação do algoritmo HubISC, que considera informações sobre hubs, para a classificação de fluxos de dados de imagens. O algoritmo HubISC foi comparado em relação aos algoritmos CPE e NCM, para a classificação de fluxos de dados de imagens. Por meio da análise experimental realizada observou-se que o algoritmo HubISC apresentou resultados superiores em termos de eficácia, além de demandar uma menor dependência de instâncias de dados rotuladas, em relação aos algoritmos comparados, o que demonstrou o potencial do algoritmo desenvolvido para a classificação de fluxos de dados de imagens. Além disso, é importante destacar que o recurso de simulador de fluxos de dados, presente no *framework* EVISClass, descrito no Capítulo 6, foi considerado para a realização dos experimentos.

Conforme descrito no Capítulo 1, a questão de pesquisa considerada nessa tese foi: *como os aspectos inerentes de cenários de aplicações reais devem ser abordados no desenvolvimento de algoritmos de classificação de fluxos de dados de imagens, para que esses algoritmos sejam capazes de produzir resultados eficazes?* A partir das investigações realizadas nos Capítulos 5, 6 e 7 foi possível observar que os algoritmos de classificação de fluxos de dados de imagens devem ser capazes de trabalhar com a ocorrência dos fenômenos

concept-drift e *concept-evolution*, considerar a disponibilidade limitada de rótulos e sua obtenção com atraso, e devem levar em consideração que, geralmente, trata-se de um contexto de alta dimensionalidade. Destaca-se que foi identificado durante os experimentos que a latência possui forte influência no desempenho preditivo dos algoritmos. Além disso, por meio desse estudo, demonstrou-se que os algoritmos de classificação de fluxos de dados de imagens podem utilizar técnicas de aprendizado ativo, principalmente em ambientes com disponibilidade limitada de um usuário especialista para o fornecimento de rótulos, o que pode permitir a obtenção de um desempenho preditivo interessante, mesmo com um menor número de instâncias de dados rotuladas. Além disso, observou-se que a utilização de *hubs* mostrou potencial na classificação de fluxos de dados de imagens, pois além de ser um aspecto da alta dimensionalidade, pode contribuir para o aprendizado ativo e também ser considerada como uma estrutura de sumarização, demonstrando novos recursos para a literatura científica da área.

8.1 Principais contribuições

As principais contribuições do trabalho descrito nesta tese são as seguintes:

- **Avaliação da influência da quantidade de imagens e de classes de imagens utilizadas para a construção de descritores de características no desempenho preditivo de algoritmos de fluxos de dados de imagens**: os estudos experimentais realizados para essa avaliação têm a capacidade de apoiar a descrição de características na classificação de fluxos de dados de imagens, observando principalmente que o desempenho preditivo dos algoritmos de classificação de fluxos de dados de imagens pode ser impactado pela qualidade da representação das imagens. Além disso, a utilização de *feature-evolution* se mostra um recurso interessante para esse contexto.
- **Desenvolvimento e disponibilização[1] para a comunidade de um *framework* de avaliação de classificadores de fluxos de dados de imagens, que considera aspectos presentes em cenários de aplicações reais**: o *framework* EVISClass é capaz de avaliar diferentes aspectos do contexto de fluxos de dados de imagens, por exemplo, *concept-drift*, *concept-evolution* e latência. Além disso, o *framework* foi desenvolvido de maneira genérica, sendo capaz de suportar diferentes configurações: algoritmos de classificação, representações das imagens, parâmetros de latência, estratégias de aprendizado ativo e medidas de avaliação. Dessa forma, por meio do *framework* EVISClass é possível padronizar a forma de avaliar algoritmos de classificação de fluxos de dados de imagens e também apoiar o aperfeiçoamento desses algoritmos para a execução em cenários de aplicações reais. Outra contribuição

[1] https://github.com/EVISClass/EVISClass

relacionada ao *framework* EVISClass é o desenvolvimento e a disponibilização do simulador de fluxos de dados (veja Figura 11).

- **Desenvolvimento e disponibilização[2] para a comunidade de um classificador eficaz para fluxos de dados de imagens, que considera aspectos presentes em cenários de aplicações reais**: o algoritmo HubISC apresentou novos recursos para a classificação de fluxos de dados de imagens. A utilização de *hubs* se mostrou uma alternativa para sumarização de instâncias e representação de classes, de maneira semelhante ao tradicional uso de centróides e medóides. Além disso, os *hubs* se mostraram eficazes como estratégia de aprendizado ativo, com uma menor quantidade de rótulos solicitados e com desempenho preditivo superior nos experimentos realizados com o algoritmo HubISC. Além disso, destaca-se que o algoritmo HubISC é capaz de lidar com os fenômenos *concept-evolution* e *concept-drift*, com a ocorrência de latência e possui parâmetros que podem influenciar na quantidade de rótulos solicitados.

8.2 Trabalhos futuros

Com a conclusão do trabalho descrito nesta tese, pode-se explorar outras propostas de trabalhos futuros, visando principalmente a incorporação de novos recursos no algoritmo HubISC. A seguir são listadas algumas destas propostas que merecem destaque:

- **Considerar *feature-evolution* na representação de imagens no algoritmo HubISC**: o conceito de *feature-evolution* não foi explorado nos experimentos com o algoritmo HubISC. Conforme apresentado na Seção 6.2.2, devido aos fenômenos *concept-drift* e *concept-evolution*, em ambientes de fluxos de dados é importante evoluir o modelo de decisão e também a representação das imagens. Os experimentos descritos na Seção 6.2.2 mostraram que um algoritmo (REBUFFI et al., 2017) que considera *feature-evolution* apresentou menor degradação dos resultados, em termos de eficácia, já que foi capaz de atualizar a representação das imagens com as novas instâncias disponíveis no fluxo de imagens. Dessa forma, pode ser interessante incorporar no algoritmo HubISC recursos de *feature-evolution* e comparar os resultados de experimentos com a abordagem atual.
- **Utilizar diferentes técnicas para a simulação do fenômeno *concept-drift***: durante os experimentos realizados nesse trabalho o fenômeno *concept-drift* foi simulado considerando a apresentação sequencial das instâncias de dados de cada classe, sendo as instâncias mais próximas do centróide da classe apresentadas primeiramente, conforme apresentado na Figura 10. No contexto de imagens outras alternativas também podem ser consideradas, como é o caso da utilização de *data augmentation* (SHORTEN; KHOSHGOFTAAR, 2019).

[2] https://github.com/EVISClass/HubISC

- **Comparar outras técnicas para representação de imagens**: baseando-se nos trabalhos correlatos, foram considerados BoVW e CNN para a descrição das imagens utilizadas nos experimentos. As arquiteturas *VGG16*, *DenseNet* e *Resnet* foram utilizadas. Outras arquiteturas de CNN também podem ser capazes de apresentar resultados interessantes, por exemplo: *LeNet, AlexNet e GoogLeNet* (GU et al., 2018). Além disso, nas CNNs podem ser considerados os vetores de características de diferentes camadas, o que pode impactar em diferentes resultados dos algoritmos de classificação de fluxos de dados de imagens.
- **Comparar o algoritmo HubISC com outros algoritmos de classificação de fluxos de dados de imagens que usam uma única arquitetura para descrição de imagens e classificação**: algumas estratégias para a classificação de fluxos de dados de imagens consideram o desenvolvimento de algoritmos que utilizam uma única arquitetura de CNN para a representação das imagens e para a classificação. Essas estratégias têm o objetivo de eliminar a necessidade de possuir dois ambientes: um para a representação das imagens e outro para a classificação. Os algoritmos *End-to-End* (CASTRO et al., 2018), IL2M (BELOUADAH; POPESCU, 2019) e OCNN (ZHENG; WEN, 2022) são exemplos dessas estratégias e mostraram resultados interessantes, em termos de eficácia, comparados aos algoritmos NCM e iCaRL nos resultados divulgados pelos autores. Dessa forma, é interessante avaliar o algoritmo HubISC em relação a esses algoritmos e analisar a viabilidade da incorporação desses conceitos.
- **Aperfeiçoar a eficiência do cálculo dos *hubs***: algoritmos baseados em *hubs* podem apresentar limitações em relação a eficiência, considerando a necessidade do cálculo da pontução *hubness*. Esse cálculo pode apresentar complexidade computacional $O(n^2)$, considerando n a quantidade de instâncias de dados. Dessa forma, pode ser interessante considerar a implementação de técnicas capazes de aperfeiçoar esse cálculo, de maneira semelhante a métodos que melhoram o cálculo de vizinhos mais próximos, como é o caso da técnica StreamBPF (*Boundary Point Factor*) descrita em (KHALIQUE; KITAGAWA; AMAGASA, 2023).
- **Considerar outros conjuntos de imagens com ruído em experimentos**: algoritmos que consideram *hubs* podem ser capazes de trabalhar em conjuntos de dados com ruído, pois o cálculo da pontuação *hubness* permite a identificação de instâncias de dados distantes das demais, o que pode sugerir a presença de ruídos. Os experimentos realizados na Seção 7.2.4 consideram o conjunto de imagens CIFAR-10 com diferentes níveis de ruído e foi possível observar o potencial do algoritmo HubISC em relação ao algoritmo CPE nessas condições. Mas pode ser importante analisar outros conjuntos de imagens e outros níveis de ruído para essa comparação, inclusive com conjuntos de dados sintéticos, conforme o estudo realizado em (TOMASEV et al., 2014) para a tarefa de agrupamento de dados.

- **Aperfeiçoar a estratégia de aprendizado ativo que considera custo de obtenção do rótulo de uma instância de dados**: os experimentos apresentados na Tabela 20 compararam diferentes estratégias de aprendizado ativo no algoritmo HubISC. Entre essas estratégias, observou-se o potencial do método baseado em custo para obtenção do rótulo. A utilização dessa estratégia também foi abordada em (PARREIRA; PRATI, 2019) e o seu potencial também foi apresentado. Entretanto, sua primeira implementação no algoritmo HubISC exige conhecer os rótulos de instâncias de dados vizinhas na fase online do algoritmo, o que pode ser impraticável em um ambiente real. Dessa forma, pelo potencial demonstrado pela estratégia observa-se a importância da investigação de outras formas mais eficientes de sua implementação no algoritmo HubISC.

8.3 Contribuições em produção bibliográfica

A seguir são apresentadas as contribuições bibliográficas produzidas durante a realização deste trabalho

- **Criação do *framework* de avaliação EVISClass**: descrita no Capítulo 6. O desenvolvimento do *framework* EVISClass gerou a seguinte publicação:

  > *M. C. de Lima, M. C. N. Barioni, E. R. Faria and H. L. Razente, "EVISClass: a new evaluation method for image data stream classifiers,"International Conference on Machine Learning and Applications (ICMLA), 2020, pp. 399-406, doi: 10.1109/ICMLA51294.2020.00070.*

- **Estudos experimentais sobre descritores de características de imagens**: descritos no Capítulo 5 e na Seção 6.2.2. O estudo experimental realizado no Capítulo 5 foi descrito no seguinte artigo:

  > *M. C. de Lima, A.J.S de Abreu, E. R. Faria and M. C. N. Barioni. "Evaluating the Construction of Feature Descriptors in the Performance of the Image Data Stream Classification". Iberoamerican Congress on Pattern Recognition (CIARP), 2021, pp: 327-339, doi: 10.1007/978-3-030-93420-0_31.*

  O estudo experimental realizado na Seção 6.2.2, que incorpora o *framework* EVISClass na análise, foi descrito no seguinte artigo:

  > *M. C. de Lima, Y. S. e Souza, E. R. Faria and M. C. N. Barioni. "A comprehensive analysis of the diverse aspects inherent to image data stream classification". Knowledge and Information Systems 64, 2215–2238 (2022), doi: 10.1007/s10115-022-01717-1.*

- **Criação do algoritmo HubISC**: descrita no Capítulo 7. Os resultados iniciais do algoritmo HubISC foram descritos na seguinte publicação:

M. C. de Lima, E. R. Faria and M. C. N. Barioni. "HubISC: um novo algoritmo baseado em hubness para a classificação de fluxo de dados de imagens". Simpósio Brasileiro de Bancos de Dados (SBBD), 2022, pp. 138-150. Búzios. DOI: 10.5753/sbbd.2022.224318.

Referências

AGRAHARI, S.; SINGH, A. K. Concept drift detection in data stream mining : A literature review. **JKSUCI**, 2021. DOI: 10.1016/j.jksuci.2021.11.006.

AKCAY, S. et al. Using deep convolutional neural network architectures for object classification and detection within x-ray baggage security imagery. **TIFS**, v. 13, n. 9, p. 2203–2215, 2018. DOI: 10.1109/TIFS.2018.2812196.

ALZUBAIDI, L. et al. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. **Journal of Big Data**, v. 8, 2021. DOI: 10.1186/s40537-021-00444-8.

ANDREU, Y.; MOLLINEDA, R.; GARCíA-SEVILLA, P. Gender recognition from a partial view of the face using local feature vectors. In: **IbPRIA**. Povoa de Varzim, Portugal: 2009. p. 481–488. DOI: 10.1007/978-3-642-02172-5_62.

ANUSHA, A. V. et al. Facial expression recognition and gender classification using facial patches. In: **COMSNETS**. Bangalore, India: 2016. p. 200–204. DOI: "10.1109/CSN.2016.7824014.

AVILA, S. et al. Bossa: Extended bow formalism for image classification. In: **ICIP**. Brussels, Belgium: 2011. p. 2909–2912. DOI: 10.1109/ICIP.2011.6116268.

AVILA, S. et al. Pooling in image representation: The visual codeword point of view. **CVIU**, v. 117, n. 5, p. 453 – 465, 2013. DOI: 10.1016/j.cviu.2012.09.007.

AWAIS, M. et al. Real-time surveillance through face recognition using hog and feedforward neural networks. **Access**, v. 7, p. 121236–121244, 2019. DOI: 10.1109/ACCESS.2019.2937810.

BELOUADAH, E.; POPESCU, A. Il2m: Class incremental learning with dual memory. In: **ICCV**. Seoul, Korea: 2019. p. 583–592. DOI: 10.1109/ICCV.2019.00067.

BENBRAHIM, H.; BEHLOUL, A. Fine-tuned xception for image classification on tiny imagenet. In: **AI-CSP**. El Oued, Algeria: [s.n.], 2021. p. 1–4. DOI: 10.1109/AI-CSP52968.2021.9671150.

BIFET, A.; FRANK, E. Sentiment knowledge discovery in twitter streaming data. In: **Discovery Science**. Canberra, Australia: 2010. p. 1–15. DOI: 10.1007/978-3-642-16184-1_1.

BIFET, A. et al. **Machine Learning for Data Streams with Practical Examples in MOA**. MIT Press, 2018. <https://moa.cms.waikato.ac.nz/book/>.

BIFET, A.; HOLMES, G.; PFAHRINGER, B. Leveraging bagging for evolving data streams. In: **ECML PKDD**. Athens, Greece: 2010. p. 135–150. DOI: 10.1007/978-3-642-15880-3_15.

BIFET, A. et al. Efficient online evaluation of big data stream classifiers. In: **SIGKDD**. Sydney, Australia: 2015. p. 59–68. DOI: 10.1145/2783258.2783372.

BIFET, A.; READ, J. Ubiquitous artificial intelligence and dynamic data streams. In: **DEBS**. Hamilton, New Zealand: 2018. p. 1–6. DOI: 10.1145/3210284.3214345.

BIFET, A. et al. Pitfalls in benchmarking data stream classification and how to avoid them. In: **ECML PKDD**. Prague, Czech Republic: 2013. p. 465–479. DOI: 10.1007/978-3-642-40988-2_30.

BO, C.; LU, H.; WANG, D. Spectral-spatial k-nearest neighbor approach for hyperspectral image classification. **Multimedia Tools Appl.**, v. 77, n. 9, p. 10419–10436, 2018. DOI: 10.1007/s11042-017-4403-9.

BOUALLEG, Y.; FARAH, M. Enhanced interactive remote sensing image retrieval with scene classification convolutional neural networks model. In: **IGARSS**. Valencia, Spain: 2018. p. 4748–4751. DOI: 10.1109/IGARSS.2018.8518388.

BOUROUIS, S. et al. Deriving probabilistic SVM kernels from flexible statistical mixture models and its application to retinal images classification. **Access**, v. 7, p. 1107–1117, 2019. DOI: 10.1109/ACCESS.2018.2886315.

CAO, Z. et al. An experimental study on breast lesion detection and classification from ultrasound images using deep learning architectures. **BMC Medical Imaging**, v. 19, n. 1, p. 51:1–51:9, 2019. DOI: 10.1186/s12880-019-0349-x.

CASTRO, F. M. et al. End-to-end incremental learning. In: **ECCV**. Munich, Germany: 2018. p. 241–257. DOI: 10.1007/978-3-030-01258-8_15.

CAVALCANTI, D. M. **Aprendizado Ativo para Classificadores de Fluxo de Dados Baseados em Agrupamento**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 10 2021.

CHAKI, J.; DEY, N. **A Beginner's Guide to Image Pre-processing Techniques**. 2018. DOI: 10.1201/9780429441134.

CHATFIELD, K. et al. The devil is in the details: An evaluation of recent feature encoding methods. In: **BMVC**. Dundee, UK: 2011. v. 2, p. 76.1–76.12. DOI: 10.5244/C.25.76.

COLLINS, J.; OKADA, K. A comparative study of similarity measures for content-based medical image retrieval. In: **CLEF**. Rome, Italy: 2012.

CSURKA, G. et al. Visual categorization with bags of keypoints. In: **Work Stat Learn Comput Vision, ECCV**. 2004. p. 1–22.

D., S. L. F. **Introdução a Mineração de Dados: conceitos básicos, algoritmos e aplicações**. São Paulo: Saraiva, 2016.

ESPAñOL, V. C. et al. A streaming flow-based technique for traffic classification applied to 12 + 1 years of internet traffic. **Telecommunication Systems**, v. 63, p. 191–204, 11 2015. DOI: 10.1007/s11235-015-0114-6.

FACELI, K.; LORENA, A. C.; GAMA, J. **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. 1. ed. LTC, 2011.

FARIA, E. R. et al. Novelty detection in data streams. **Artificial Intelligence Review**, v. 45, n. 2, 2016. DOI: 10.1007/s10462-015-9444-8.

FELDBAUER, A. F. R. A comprehensive empirical comparison of hubness reduction in high-dimensional spaces. **Knowl Inf Syst.**, v. 59, p. 137–166, 01 2019. DOI: 10.1007/s10115-018-1205-y.

GAMA, J. **Knowledge Discovery from Data Streams**. 1st. ed. Chapman e Hall/CRC, 2010. DOI: 10.5555/1855075.

GAMA, J.; RODRIGUES, P.; SEBASTIãO, R. Evaluating algorithms that learn from data streams. In: **ACM SAC**. Honolulu, Hawaii: 2009. p. 1496–1500. DOI: 10.1145/1529282.1529616.

GAMA, J. et al. A survey on concept drift adaptation. **ACM Comput. Surv.**, v. 46, n. 4, p. 44:1–44:37, 2014. DOI: 10.1145/2523813.

GANGINENI, S. R. et al. Real-time object recognition from streaming lidar point cloud data. In: **DEBS**. Darmstadt, Germany: 2019. p. 214–219. DOI: 10.1145/3328905.3330297.

GARCIA, K. D.; CARVALHO, A. C. P. L. F. de; MENDES-MOREIRA, J. A cluster-based prototype reduction for online classification. In: **IDEAL**. Madrid, Spain: 2018. p. 603–610. DOI: 10.1007/978-3-030-03493-1_63.

GONG, M.; SHU, Y. Real-time detection and motion recognition of human moving objects based on deep learning and multi-scale feature fusion in video. **Access**, v. 8, p. 25811–25822, 2020. DOI: 10.1109/ACCESS.2020.2971283.

GOO, W. et al. Taxonomy-regularized semantic deep convolutional neural networks. In: **ECCV**. Amsterdam, Netherlands: 2016. p. 86–101. DOI: 10.1007/978-3-319-46475-6_6.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. <http://www.deeplearningbook.org>.

GRZENDA, M.; GOMES, H. M.; BIFET, A. Delayed labelling evaluation for data streams. **Data Mining and Knowledge Discovery**, 11 2019. DOI: 10.1007/s10618-019-00654-y.

GU, J. et al. Recent advances in convolutional neural networks. **Pattern Recognition**, v. 77, p. 354–377, 2018. DOI: 10.1016/j.patcog.2017.10.013.

GURJAR, G. S.; CHHABRIA, S. A review on concept evolution technique on data stream. In: **PerCom**. Pune, India: 2015. p. 1–3. DOI: 10.1109/PERVASIVE.2015.7087172.

HAQUE, A.; KHAN, L.; BARON, M. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In: **AAAI**. Phoenix, Arizona: 2016. p. 1652–1658. DOI: 10.5555/3016100.3016130.

HAQUE, A. et al. Efficient handling of concept drift and concept evolution over stream data. In: **ICDE**. Helsinki, Finland: 2016. p. 481–492. DOI: 10.1109/ICDE.2016.7498264.

HARE, J. S.; SAMANGOOEI, S.; DUPPLAW, D. P. Openimaj and imageterrier: Java libraries and tools for scalable multimedia analysis and indexing of images. In: **ACM Multimedia**. Scottsdale, USA: 2011. p. 691–694. DOI: 10.1145/2072298.2072421.

HOU, J.; KANG, J.; QI, N. On vocabulary size in bag-of-visual-words representation. In: **PCM**. Shanghai, China: 2010. p. 414–424. DOI: 10.1007/978-3-642-15702-8_38.

HU, J. et al. Online user modeling for interactive streaming image classification. In: **MMM**. Reykjavik, Iceland: 2017. p. 293–305. DOI: 10.1007/978-3-319-51814-5_25.

HU, J. et al. Accumulative image categorization: a personal photo classification method for progressive collection. **Multimedia Tools Appl.**, v. 77, n. 24, p. 32179–32211, 2018. DOI: 10.1007/s11042-018-6152-9.

HU, J. et al. An integrated classification model for incremental learning. **Multimed Tools Appl**, v. 80, p. 17275–17290, 2021. DOI: 10.1007/s10115-023-01854-1.

HUANG, K.; WANG, C.; TAO, D. High-order topology modeling of visual words for image classification. **TIP**, v. 24, n. 11, p. 3598–3608, 2015. DOI: 10.1109/TIP.2015.2449081.

JAIN, S. et al. Object detection using coco dataset. In: **ICCR**. Dubai,UAE: [s.n.], 2022. p. 1–4. DOI: 10.1109/ICCR56254.2022.9995808.

JAISWAL, G. Performance analysis of incremental learning strategy in image classification. In: **Confluence**. Uttar Pradesh, India: 2021. p. 427–432. DOI: 10.1109/Confluence51648.2021.9377034.

JANARDAN; MEHTA, S. Concept drift in streaming data classification: Algorithms, platforms and issues. **PROCS**, v. 122, p. 804–811, 2017. DOI: 10.1016/j.procs.2017.11.440.

JIANG, L. et al. Beyond synthetic noise: Deep learning on controlled noisy labels. In: **ICML**. Vienna, Austria: 2020. DOI: 10.5555/3524938.3525384.

JUNIOR, O. L. et al. Trainable classifier-fusion schemes: An application to pedestrian detection. In: **ITSC**. St. Louis, Missouri: 2009. p. 1–6. DOI: 10.1109/ITSC.2009.5309700.

KÄDING, C. et al. Fine-tuning deep neural networks in continuous learning scenarios. In: **ACCV**. Taipei, Taiwan: 2017. p. 588–605. DOI: 10.1007/978-3-319-54526-4_43.

Karhan, Z.; Ergen, B. Content based medical image classification using discrete wavelet and cosine transforms. In: **SIU**. Malatya, Turkey: 2015. p. 1445–1448. DOI: 10.1109/SIU.2015.7130115.

KHALIQUE, V.; KITAGAWA, H.; AMAGASA, T. Bpf: a novel cluster boundary points detection method for static and streaming data. **Knowl Inf Syst**, v. 65, p. 2991–3022, 2023. DOI: 10.1007/s10115-023-01854-1.

KUMAR, M. D. et al. A comparative study of cnn, bovw and lbp for classification of histopathological images. In: **SSCI**. Honolulu, Hawaii: 2017. p. 1–7. DOI: 10.1109/SSCI.2017.8285162.

KWITT, R.; VASCONCELOS, N.; RASIWASIA, N. Scene recognition on the semantic manifold. In: **ECCV**. Florence, Italy: 2012. p. 359–372. DOI: 10.1007/978-3-642-33765-9_26.

LAZEBNIK, S.; SCHMID, C.; PONCE, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: **CVPR**. New York, USA: 2006. v. 2, p. 2169–2178. DOI: 10.1109/CVPR.2006.68.

LI, W. et al. Deep learning-based classification methods for remote sensing images in urban built-up areas. **Access**, v. 7, p. 36274–36284, 2019. DOI: 10.1109/ACCESS.2019.2903127.

LI, X. et al. Socializing the semantic gap: A comparative survey on image tag assignment, refinement, and retrieval. **ACM Computing Surveys**, v. 49, n. 1, p. 14:1–14:39, 2016. DOI: 10.1145/2906152.

LI, Z.; DING, Q.; ZHANG, W. A comparative study of different distances for similarity estimation. In: **ICICIS**. Chongqing, China: 2011. p. 483–488. DOI: 10.1007/978-3-642-18129-0_75.

LIMA, M. C. de et al. A comprehensive analysis of the diverse aspects inherent to image data stream classification. **Knowl and Inf. Systems**, p. 2215–2238, 08 2022. DOI: 10.1007/s10115-022-01717-1.

LIU, Y.-L.; CAI, Z.; ZHANG, J. An improved image classification based on k-means clustering and bow model. **IJGUC**, v. 9, p. 37–42, 2018. DOI: 10.1504/IJGUC.2018.10011388.

LOSING, V.; HAMMER, B.; WERSING, H. Knn classifier with self adjusting memory for heterogeneous concept drift. In: **ICDM**. Barcelona, Spain: 2016. p. 291–300. DOI: 10.1109/ICDM.2016.0040.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **Int. J. Comput. Vision**, v. 60, n. 2, p. 91–110, 2004. DOI: 10.1023/B:VISI.0000029664.99615.94.

LUO, L. et al. Nonparametric bayesian correlated group regression with applications to image classification. **TNNLS**, v. 29, n. 11, p. 5330–5344, 2018. DOI: 10.1109/TNNLS.2018.2797539.

MANI, P. et al. The hubness phenomenon in high-dimensional spaces. **AWMS**, p. 15–45, 03 2019. DOI: 10.1007/978-3-030-11566-1_2.

MASUD, M. M. et al. Classification and adaptive novel class detection of feature-evolving data streams. **TKDE**, v. 25, n. 7, p. 1484–1497, 2013. DOI: 10.1109/TKDE.2012.109.

MENSINK, T. et al. Distance-based image classification: Generalizing to new classes at near-zero cost. **TPAMI**, v. 35, n. 11, p. 2624–2637, 2013. DOI: 10.1109/TPAMI.2013.83.

MIEBS, G. et al. Efficient strategies of static features incorporation into the recurrent neural network. **Neural Processing Letters**, v. 51, 06 2020. DOI: 10.1007/s11063-020-10195-x.

MIRMEHDI, M.; XIE, X.; SURI, J. **Handbook of Texture Analysis**. London, UK: Imperial College Press, 2009. DOI: 10.1142/p547.

MU, X. et al. Streaming classification with emerging new class by class matrix sketching. In: **AAAI**. San Francisco, USA: 2017. p. 2373–2379. DOI: 10.5555/3298483.3298581.

NAKATA, K. et al. Revisiting a knn-based image classification system with high-capacity storage. In: **ECCV**. Cham: Springer, 2022. p. 457–474. DOI: 10.1007/978-3-031-19836-6_26.

NANDI, A. et al. Real-time emotion classification using eeg data stream in e-learning contexts. **Sensors**, v. 21, p. 1589, 02 2021. DOI: 10.3390/s21051589.

NAPOLETANO, P. Hand-crafted vs learned descriptors for color texture classification. In: **CCIW**. Milan, Italy: 2017. p. 259–271. DOI: 10.1007/978-3-319-56010-6_22.

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. **Knowl. Inf. Syst.**, v. 45, n. 3, p. 535–569, dez. 2015. DOI: 10.1007/s10115-014-0808-1.

NOVAK, C. L.; SHAFER, S. A. Anatomy of a color histogram. In: **CVPR**. Champaign, Illinois: 1992. p. 599–605. DOI: 10.1109/CVPR.1992.223129.

PARREIRA, P.; PRATI, R. Active learning in data stream with intermediate latency. In: **ENIAC**. Salvador, Brazil: 2019. DOI: 10.5753/eniac.2019.

PESTOV, V. Intrinsic dimension of a dataset: what properties does one expect? In: **IJCNN**. Orlando, Florida: 2007. p. 2959–2964. DOI: 10.1109/IJCNN.2007.4371431.

PFAHRINGER, B.; HOLMES, G.; KIRKBY, R. New options for hoeffding trees. In: **AJCAI**. Gold Coast, Australia: 2007. p. 90–99. DOI: 10.1007/978-3-540-76928-6_11.

POWERS, D.; AILAB. Evaluation: From precision, recall and f-measure to roc, informedness, markedness e correlation. **J. Mach. Learn. Technol**, v. 2, p. 2229–3981, 01 2011. DOI: 10.9735/2229-3981.

PUGLIESE, V. U.; COSTA, R. D.; HIRATA, C. M. Comparative evaluation of the supervised machine learning classification methods and the concept drift detection methods in the financial business problems. In: **ICEIS**. Virtual Conference: 2020. DOI: 10.1007/978-3-030-75418-1_13.

PULLAR-STRECKER, Z. et al. Hitting the target: stopping active learning at the cost-based optimum. **Mach Learn**, 10 2022. DOI: 10.1007/s10994-022-06253-1.

PUNN, N. S.; AGARWAL, S. Testing concept drift detection technique on data stream. In: **IBDAP**. Warangal, India: 2018. p. 89–99. DOI: 10.1007/978-3-030-04780-1_6.

RAZAVIAN, A. S. et al. Cnn features off-the-shelf: An astounding baseline for recognition. In: **CVPRW**. Columbus, Ohio: 2014. p. 512–519. DOI: 10.1109/CVPRW.2014.131.

REBUFFI, S.-A. et al. icarl: Incremental classifier and representation learning. In: **CVPR**. Honolulu, Hawaii: 2017. p. 5533–5542. DOI: 10.1109/CVPR.2017.587.

RISTIN, M. et al. Incremental learning of ncm forests for large-scale image classification. In: **CVPR**. Columbus, Ohio: 2014. p. 3654–3661. DOI: 10.1109/CVPR.2014.467.

ROSNER, B.; GLYNN, R.; LEE, M.-L. The wilcoxon signed rank test for paired comparisons of clustered data. **Biometrics**, v. 62, p. 185–92, 03 2006. DOI: 10.1111/j.1541-0420.2005.00389.x.

RZANNY, M. et al. Acquiring and preprocessing leaf images for automated plant identification: Understanding the tradeoff between effort and information gain. **Plant Methods**, v. 13, p. 97, 11 2017. DOI: 10.1186/s13007-017-0245-8.

SAAD, S. M.; KAMARUDIN, S. S. Comparative analysis of similarity measures for sentence level semantic measurement of text. In: **ICCSCE**. Penang, Malaysia: 2013. p. 90–94. DOI: 10.1109/ICCSCE.2013.6719938.

SAINI, M.; SUSAN, S. Comparison of deep learning, data augmentation and bag of-visual-words for classification of imbalanced image datasets. In: **RTIP2R**. Solapur, India: 2019. p. 561–571. DOI: 10.1007/978-981-13-9181-1_49.

SAMET, H. **Foundations of Multidimensional and Metric Data Structures**. Morgan Kaufmann, 2005.

SCHUCH, P. **Deep Learning for Fingerprint Recognition Systems**. Tese (Doutorado) — Norwegian University of Science and Technology, 10 2019.

SETTLES, B. **Active Learning Literature Survey**. University of Wisconsin, 2010.

SHI, D. et al. Batch and data streaming classification models for detecting adverse events and understanding the influencing factors. **ENGAPPAI**, v. 85, p. 72–84, 2019. DOI: 10.1016/j.engappai.2019.05.006.

SHORTEN, C.; KHOSHGOFTAAR, T. A survey on image data augmentation for deep learning. **Big Data**, v. 6, 2019. DOI: 10.1186/s40537-019-0197-0.

SILVA, F. B. et al. Graph-based bag-of-words for classification. **PATCOG**, v. 74, n. C, p. 266–285, 2018. DOI: 10.1016/j.patcog.2017.09.018.

SILVA, J. A. et al. Data stream clustering: A survey. **ACM Comput. Surv.**, v. 46, n. 1, p. 13:1–13:31, 2013. DOI: 10.1145/2522968.2522981.

SOUZA, V. M. A. et al. Classification of evolving data streams with infinitely delayed labels. In: **ICMLA**. Miami, Florida: 2015. p. 214–219. DOI: 10.1109/ICMLA.2015.174.

SRIVASTAVA, D.; BAKTHULA, R.; AGARWAL, S. Image classification using surf and bag of lbp features constructed by clustering with fixed centers. **Multimedia Tools Appl.**, v. 78, n. 11, p. 14129–14153, 2019. DOI: 10.1007/s11042-018-6793-8.

SRIVASTAVA, D.; RAJITHA, B.; Agarwal, S. An efficient image classification using bag-of-words based on surf and texture features. In: **INDICON**. Roorkee, India: 2017. p. 1–6. DOI: 10.1109/INDICON.2017.8488010.

STEFANOWSKI, J.; BRZEZINSKI, D. Stream classification. **Encyclopedia of Machine Learning and Data Mining**, p. 1191–1199, 2017. DOI: 10.1007/978-1-4899-7687-1_908.

TEZCAN, C. E.; KIRAS, B.; BILGIN, G. Classification of breast cancer histopathological images with deep transfer learning methods. In: **SIU**. Safranbolu, Turkey: 2022. p. 1–4. DOI: 10.1109/SIU55565.2022.9864846.

THOMAS, J. J.; PILLAI, N. A deep learning framework on generation of image descriptions with bidirectional recurrent neural networks. In: **ICO**. Pattaya, Thailand: 2019. p. 219–230. DOI: 10.1007/978-3-030-00979-3_22.

TOMASEV, N. et al. A probabilistic approach to nearest-neighbor classification: Naive hubness bayesian knn. In: **CIKM**. Glasgow, Scotland: 2011. p. 2173–2176. DOI: 10.1145/2063576.2063919.

TOMAŠEV, N. et al. Hubness-based fuzzy measures for high-dimensional k-nearest neighbor classification. **Int. J. Mach. Learn. e Cyber.**, v. 5, p. 445–458, 2014. DOI: 10.1007/s13042-012-0137-1.

TOMASEV, N. et al. The role of hubness in clustering high-dimensional data. **TKDE**, v. 26, n. 3, p. 739–751, 2014. DOI: 10.1109/TKDE.2013.25.

TOMASIK, B.; THIHA, P.; TURNBULL, D. Tagging products using image classification. In: **SIGIR**. Boston, USA: 2009. p. 792–793. DOI: 10.1145/1571941.1572131.

TOMAšEV, N.; BUZA, K. Hubness-aware knn classification of high-dimensional data in presence of label noise. **Neurocomputing**, v. 160, p. 157–172, 2015. ISSN 0925-2312. DOI: 10.1016/j.neucom.2014.10.084.

VELOSO, G. **Proposta de Bag-of-Visual-Words por meio de Redes Complexas**. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, 10 2017.

WANG, H. et al. Feature selection for image classification based on bacterial colony optimization. In: **ICSI**. Qingdao, China: 2021. p. 430–439. DOI: 10.1007/978-3-030-78811-7_40.

WANG, K. et al. Cost-effective active learning for deep image classification. **TCSVT**, v. 27, n. 12, p. 2591–2600, 2017. DOI: 10.1109/TCSVT.2016.2589879.

WANG, Z. et al. Robust high dimensional stream classification with novel class detection. In: **ICDE**. Macao, Macao: 2019. p. 1418–1429. DOI: 10.1109/ICDE.2019.00128.

WILLIAMS, T.; LI, R. Advanced image classification using wavelets and convolutional neural networks. In: **ICMLA**. Anaheim, California: 2016. p. 233–239. DOI: 10.1109/ICMLA.2016.0046.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining: Practical Machine Learning Tools and Techniques**. 3. ed. Morgan Kaufmann, 2011.

WU, J. et al. Multi-label active learning algorithms for image classification: Overview and future promise. **ACM Comput. Surv.**, v. 53, n. 2, 2020. DOI: 10.1145/3379504.

WU, Q. et al. Hiboost: A hubness-aware ensemble learning algorithm for high-dimensional imbalanced data classification. **JIFS**, v. 39, p. 1–12, 06 2020. DOI: 10.3233/JIFS-190821.

WU, Y. et al. Large scale incremental learning. In: **CVPR**. Long Beach, California: 2019. p. 374–382. DOI: 10.1109/CVPR.2019.00046.

YU, A. et al. A novel framework for face recognition using robust local representation-based classification. **IJDSN**, v. 15, n. 3, 2019. DOI: 10.1177/1550147719836082.

ZHENG, T.; WEN, Z. Online convolutional neural network for image streams classification. In: **ICBDT**. Qingdao, China: ACM, 2022. p. 255–259. DOI: 10.1145/3565291.3565332.

ZHU, X. et al. Active learning from stream data using optimal weight classifier ensemble. **TSMCB**, v. 40, n. 6, p. 1607–1621, 2010. DOI: 10.1109/TSMCB.2010.2042445.

ZHUANG, J. et al. Deep knn for medical image classification. In: **MICCAI**. Lima, Peru: 2020. p. 127–136. DOI: 10.1007/978-3-030-59710-8_13.

ŽLIOBAITĖ, I. et al. Active learning with evolving streaming data. In: **ECML PKDD**. Athens, Greece: 2011. p. 597–612. DOI: 10.1007/978-3-642-23808-6_39.

ŽLIOBAITė, I. et al. Active learning with drifting streaming data. **TNNLS**, v. 25, n. 1, p. 27–39, 2014. DOI: 10.1109/TNNLS.2012.2236570.