



ALEXP: UM FRAMEWORK EXPERIMENTAL DE APRENDIZADO ATIVO

Felipe Gomes Táparo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Miguel Elias Mitre Campista

Rio de Janeiro
Fevereiro de 2026

ALEXP: UM FRAMEWORK EXPERIMENTAL DE APRENDIZADO ATIVO

Felipe Gomes Táparo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Miguel Elias Mitre Campista

Aprovada por: Prof. Miguel Elias Mitre Campista

Prof. Rodrigo de Sousa Couto

Prof. Igor Monteiro Moraes

RIO DE JANEIRO, RJ – BRASIL

FEVEREIRO DE 2026

Gomes Táparo, Felipe

ALExp: Um Framework Experimental de Aprendizado Ativo/Felipe Gomes Táparo. – Rio de Janeiro: UFRJ/COPPE, 2026.

XIV, 37 p.: il.; 29, 7cm.

Orientador: Miguel Elias Mitre Campista

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2026.

Referências Bibliográficas: p. 35 – 37.

1. Aprendizado Ativo. 2. Redes Neurais. I. Elias Mitre Campista, Miguel. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Aos meus pais.

Agradecimentos

Agradeço aos meus pais, por sempre terem me apoiado e me amado incondicionalmente, e por incentivarem a seguir meus interesses e minhas ambições.

Agradeço aos meus amigos, os quais posso contar em todas as situações.

Agradeço à minha namorada, por ser minha companheira fiel e estar ao meu lado em todos os momentos.

Agradeço aos meus professores por despertarem meu interesse e me fornecerem o conhecimento necessário na minha formação.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALEXP: UM FRAMEWORK EXPERIMENTAL DE APRENDIZADO ATIVO

Felipe Gomes Táparo

Fevereiro/2026

Orientador: Miguel Elias Mitre Campista

Programa: Engenharia Elétrica

O sucesso no treinamento de modelos supervisionados de aprendizado de máquina depende de grandes quantidades de dados rotulados. No entanto, a tarefa de anotar uma grande massa de dados é custosa e demorada. A fim de amenizar o esforço de rotulação, técnicas de Aprendizado Ativo buscam selecionar as amostras não rotuladas mais representativas, de forma a focar os esforços de rotulação em amostras que darão maior ganho de desempenho ao modelo.

Este trabalho apresenta o *Hybrid Strategy K-Center – HSKC*, uma nova estratégia híbrida para a realização de Aprendizado Ativo. O HSKC mistura heurísticas clássicas em um cenário sensível ao tamanho do *batch* de seleção, ainda que essas heurísticas não sejam sensíveis originalmente.

Este trabalho também apresenta o ALExp, um *framework* experimental para aprendizado ativo. Com a ferramenta proposta, é possível realizar simulações de cenários de aprendizado ativo de forma robusta e simples. O ALExp abstrai o gerenciamento do ciclo de vida de experimentos de aprendizado ativo, oferecendo uma abstração simples e eficiente para dividir *datasets*, simular rotulação e aplicar transformações. Uma interface simplificada para a implementação de novas estratégias é fornecida. O *framework* também implementa diversas estratégias de seleção presentes na literatura. Ademais, a ferramenta conta com métodos para a aplicação de aprendizado ativo em cenários práticos.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ALEXP: A ACTIVE LEARNING EXPERIMENTAL FRAMEWORK

Felipe Gomes Táparo

February/2026

Advisor: Miguel Elias Mitre Campista

Department: Electrical Engineering

Successful training of supervised machine learning models depends on large amounts of labeled data. However, the task of annotating a massive amount of data is costly and time-consuming. To mitigate the labeling effort, Active Learning techniques aim to select the most representative unlabeled samples, thereby focusing labeling efforts on the instances that will yield the greatest performance gain for the model.

This work presents the *Hybrid Strategy K-Center – HSKC*, a novel hybrid strategy for performing Active Learning. HSKC blends classical heuristics in a scenario that is sensitive to the selection batch size, even though these heuristics are not originally sensitive to it.

This work also introduces ALExp, an experimental framework for active learning. With the proposed tool, it is possible to run simulations of active learning scenarios robustly and simply. ALExp abstracts the lifecycle management of active learning experiments, providing a straightforward and efficient abstraction for splitting datasets, simulating labeling, and applying transformations. A simplified interface for implementing new strategies is provided. The framework also implements several selection strategies found in the literature. Furthermore, the tool features methods for applying active learning in practical scenarios.

Sumário

Lista de Figuras	x
Lista de Tabelas	xi
Lista de Códigos	xiii
1 Introdução	1
2 Fundamentação Teórica	4
2.1 Aprendizado Ativo	4
2.1.1 Técnicas Clássicas de Aprendizado Ativo	6
2.1.2 BALD	7
2.1.3 Coreset	8
2.1.4 BADGE	9
3 Hybrid Strategy K-Center	11
3.1 Proposta	11
3.1.1 Poda Bilateral	12
3.2 Influência	12
3.3 Sobre o Consumo de Memória	12
4 ALExp: Um <i>Framework</i> de Aprendizado Ativo	14
4.1 Motivação	14
4.1.1 Estratégias	14
4.1.2 <i>Datasets</i>	16
4.1.3 Utilidades	19
4.2 Laço de aprendizado ativo com a ferramenta	19
5 Experimentos em Aprendizado Ativo	21
5.1 Procedimentos Experimentais	21
5.2 Geração do <i>Datasets</i> Redundantes	23
5.3 Experimentos	26
5.3.1 Experimentos em pequena escala com <i>datasets</i> clássicos	26

5.3.2	Experimentos com CIFAR-10 e CIFAR-10 redundante	28
5.3.3	Experimentos com MNIST redundante	31
5.4	Conclusões Sobre os Experimentos	33
6	Conclusão	34
	Referências Bibliográficas	35

Lista de Figuras

2.1	Laço de seleção de dados no aprendizado ativo.	5
4.1	Visão geral do funcionamento do ALExp.	15
4.2	Implementação das estratégias no <i>framework</i>	17
4.3	Diagrama UML das classes implementadas no arquivo <code>datasets</code>	18
4.4	Laço de aprendizado ativo de um experimento com o arcabouço desenvolvido.	20
5.1	Distribuição de classes no <i>dataset</i> CIFAR-10 redundante gerado, com fator de redundância 5.	25
5.2	Distribuição de classes no <i>dataset</i> MNIST redundante gerado, com fator de redundância 5.	25
5.3	Teste de estratégias no SVHN.	27
5.4	Teste de estratégias no MNIST.	27
5.5	Teste de estratégias no CIFAR10.	29
5.6	Teste de estratégias no CIFAR10 redundante.	29
5.7	Teste de estratégias no CIFAR-10 Redundante, com número de amostras iniciais reduzidas.	30
5.8	Teste de estratégias no CIFAR10 redundante em grande escala.	31
5.9	Teste de estratégias no MNIST redundante.	32
5.10	Teste de estratégias no MNIST redundante, com número de amostras iniciais reduzidas.	32

Lista de Tabelas

3.1	Estimativa de Memória para Seleção ($N = 50k$, $D = 1024$, $C = 100$).	13
5.1	Transformações aplicadas aos <i>Datasets</i> .	24
5.2	Distribuição de amostras e originais utilizadas por classe no <i>dataset</i> gerado.	26
5.3	Distribuição de amostras e originais utilizadas por classe no <i>dataset</i> gerado.	26
5.4	Comparação da Acurácia de Teste (%) ao Longo dos Ciclos de Aprendizado Ativo no <i>dataset</i> SHVN.	27
5.5	Comparação da Acurácia de Teste (%) ao Longo dos Ciclos de Aprendizado Ativo no <i>dataset</i> MNIST.	28
5.6	Comparação da Acurácia de Teste (%) no CIFAR-10 ao Longo dos Ciclos de Aprendizado Ativo.	29
5.7	Comparação da Acurácia de Teste (%) no CIFAR-10 Redundante ao Longo dos Ciclos de Aprendizado Ativo.	30
5.8	Comparação da Acurácia de Teste (%) no CIFAR-10 Redundante em Escala Reduzida ao Longo dos Ciclos de Aprendizado Ativo.	30
5.9	Comparação da Acurácia de Teste (%) no CIFAR-10 Redundante em Grande Escala ao Longo dos Ciclos de Aprendizado Ativo.	31
5.10	Comparação da Acurácia de Teste (%) no MNIST Redundante ao Longo dos Ciclos de Aprendizado Ativo.	32
5.11	Comparação da Acurácia de Teste (%) no MNIST Redundante em Escala Reduzida ao Longo dos Ciclos de Aprendizado Ativo.	33

Lista de Algoritmos

1	BADGE: Aprendizado Ativo em Lote via Embeddings de Gradiente Diversos.	10
2	Procedimento experimental.	22

Lista de Códigos

5.1	Implementação da ResNet18 com Dropout, utilizando Pytorch e a Resnet do Torchvision.	22
-----	--	----

Capítulo 1

Introdução

Modelos supervisionados de aprendizado de máquina são treinados a partir de um conjunto de dados rotulados, isto é, um conjunto de dados onde, para cada amostra, existe uma informação anotada por um humano sobre qual deveria ser a resposta do modelo para aquela amostra. No entanto, modelos de aprendizado de máquina frequentemente necessitam de muitos dados para generalizar de forma satisfatória. Conjuntos de dados da literatura, como o CIFAR-10 e o CIFAR-100 KRIZHEVSKY (2009) possuem 60.000 amostras rotuladas, enquanto o ImageNet DENG *et al.* (2009) supera 14 milhões de imagens rotuladas. Rotular essa quantidade massiva de dados é frequentemente muito custoso e demorado.

A fim de contornar o problema de escassez de rótulos, abordagens como o *Pseudo-labeling* e o Aprendizado Ativo (*Active Learning – AL*) buscam mitigar o alto custo de anotação supervisionada. O *Pseudo-labeling* é uma técnica de aprendizado supervisionado que propõe adicionar rótulos provisórios a amostras não rotuladas. Uma abordagem comum em *Pseudo-labeling* consiste em, primeiramente, treinar classificadores em dados rotulados e, em seguida, utilizar as previsões dos classificadores resultantes para gerar dados rotulados adicionais VAN ENGELEN e HOOS (2020). Por outro lado, o Aprendizado Ativo é a área de estudos que propõe técnicas para selecionar as amostras mais informativas de um grande conjunto de dados não rotulados. O objetivo do Aprendizado Ativo é selecionar as amostras com maior impacto no modelo treinado, sendo essas amostras, então, rotuladas por um oráculo humano para serem utilizadas no treinamento.

A realização de experimentos com Aprendizado Ativo requer implementar estratégias e gerenciar o ciclo de vida de cada experimento. Para a experimentação e teste de estratégias, a abordagem padrão é utilizar um conjunto de dados rotulados, remover os rótulos artificialmente e utilizar uma estratégia de seleção para adicionar os rótulos gradualmente SENNER e SAVARESE (2018); ASH *et al.* (2020), conforme a estratégia indica as amostras mais representativas. No entanto, implementar essas estratégias e gerenciar o experimento de forma robusta não é uma tarefa trivial. Na

literatura, existem algumas ferramentas que propõem a implementação de um *framework* experimental de Aprendizado Ativo, como o Baal ATIGHEHCHIAN *et al.* (2022) e o ModAL DANKA e HORVATH. No entanto, o Baal foca especificamente em Aprendizado Ativo Bayesiano LI *et al.* (2025), enquanto ModAL é uma biblioteca modular, e não implementa estratégias mais complexas, como o BALD HOULSBY *et al.* (2011), o Coreset SENER e SAVARESE (2018) e o BADGE ASH *et al.* (2020).

Este trabalho apresenta o ALExp, um *framework* de aprendizado ativo que busca implementar as principais técnicas da literatura em um ambiente experimental. A ferramenta provê um ambiente que simula o ciclo de vida de experimentos, utilizando *datasets* já rotulados e removendo seus rótulos artificialmente, sendo rotulados novamente a critério de estratégias de seleção. O ALExp implementa as principais heurísticas (Baixa confiança, Entropia, Margem e Similaridade de *Embedding*) e estratégias da literatura (BALD, Coreset e BAGDE). Este trabalho também propõe e implementa o *Hybrid Strategy K-Center – (HSKC)*, uma nova estratégia de seleção mista que combina múltiplas heurísticas clássicas em uma única estratégia. Essa estratégia busca misturar heurísticas clássicas de uma maneira sensível ao *batch*, enquanto mantém uma pegada de memória reduzida, diferente de outras estratégias sensíveis ao *batch*, como o BADGE e o Coreset. A fim de demonstrar as estratégias implementadas, foram realizados experimentos com os *datasets* MNIST DENG (2012), SVHN NETZER *et al.* (2011) e CIFAR-10.

O trabalho está estruturado da seguinte maneira:

- O Capítulo 2 revisa a literatura de aprendizado ativo, introduzindo o problema, apresentando heurísticas clássicas e estratégias consideradas como estado da arte pela literatura. Este capítulo também apresenta a teoria de aprendizado por reforço, útil para compreender algumas seções do texto.
- O Capítulo 3 apresenta o *Hybrid Strategy K-Center – (HSKC)*, uma nova estratégia híbrida desenvolvida para ser utilizada em conjunto com heurísticas clássicas. O HSKC possui um desempenho satisfatório nos testes realizados, enquanto mantém uma baixa utilização de memória.
- O Capítulo 4 apresenta o ALExp, um *framework* experimental de Aprendizado Ativo. Essa parte do texto detalha sua implementação e como a ferramenta pode ser utilizada para testar e desenvolver novas estratégias, abstraindo o ciclo de vida experimental. Vale ressaltar que esse *framework* implementa por padrão todas as estratégias apresentadas no Capítulo 2.
- O Capítulo 5 apresenta o resultado de testes experimentais em *datasets* clássicos da literatura, utilizando o ALExp.

- O Capítulo 6 conclui o trabalho, revisando as contribuições observadas e os resultados obtidos.

Capítulo 2

Fundamentação Teórica

Este capítulo revisará a teoria de Aprendizado Ativo, iniciando pelo conceito, a formulação, passando pelos tipos de aprendizado ativo e citando finalmente as principais técnicas. Primeiramente, serão explicadas as técnicas clássicas e, em sequência, as estratégias sensíveis a *batch*. Este capítulo é fundamental para o entendimento das técnicas implementadas no ALExp, bem como para o entendimento da formulação e do embasamento teórico da realização de Aprendizado Ativo.

2.1 Aprendizado Ativo

O sucesso de modelos de aprendizado de máquina treinados de forma supervisionada depende da existência de um conjunto de dados anotados por humanos. Por exemplo, em tarefas de classificação de imagem, o conjunto de dados rotulados tipicamente atinge as dezenas de milhares de amostras. No entanto, a tarefa de anotar dados continuamente é extremamente demorada, custosa e trabalhosa LI *et al.* (2025). O Aprendizado Ativo busca reduzir o esforço para a anotação de dados, selecionando os dados não rotulados mais relevantes para o treinamento de um modelo. O aprendizado ativo envolve, primeiramente, selecionar algumas instâncias de um conjunto de dados não rotulado, as quais são então anotadas por um oráculo humano. Esse processo é repetido diversas vezes até que um critério de parada seja satisfeito, como, por exemplo, o esgotamento do orçamento de anotação FANG *et al.* (2017).

O processo de aprendizado ativo pode ser definido como contendo um conjunto de amostras sem rótulos D_u , um conjunto de dados sem rótulos selecionados para a anotação D_n , onde D_n é um subconjunto de D_u , e um conjunto de dados rotulados D_l . Os dados em D_u são selecionados para rotulação a partir de uma estratégia de amostragem $f(x_i, \theta)$, onde x_i é um dado presente em D_u e θ se refere aos parâmetros atuais do modelo que está sendo treinado. A Figura 2.1 ilustra o laço de seleção e rotulação.

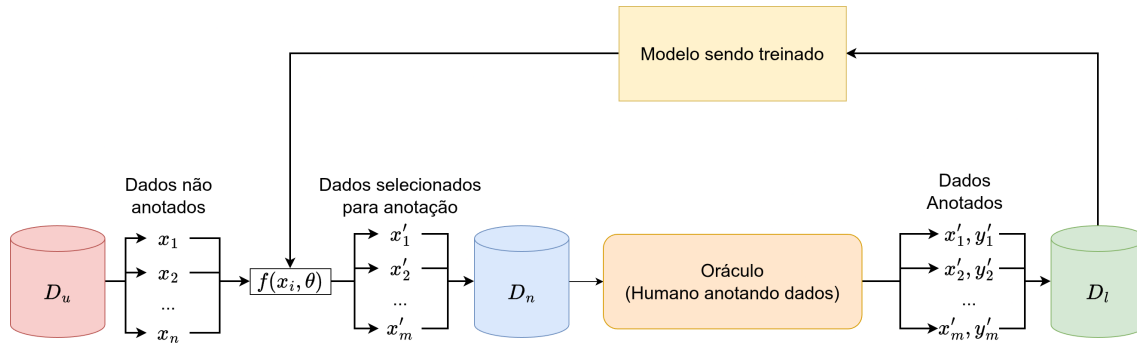


Figura 2.1: Laço de seleção de dados no aprendizado ativo.

Existem diversas estratégias de seleção de dados no aprendizado ativo. Em LI *et al.* (2025), os autores dividem as estratégias em cinco tipos:

- **Métodos baseados em incerteza:** buscam escolher os exemplos mais ambíguos conforme as previsões do modelo. Os métodos baseados em incerteza focam no desenvolvimento de diversas funções de pontuação para inferir a incerteza de uma amostra. Exemplos de métodos baseados em incerteza envolvem entropia preditiva XIE *et al.* (2022) e seleção de amostras de baixa confiança WANG e SHANG (2014). Frequentemente, esses métodos sofrem de algumas desvantagens, como a seleção de amostras redundantes pouco representativas.
- **Métodos baseados em representatividade:** buscam selecionar as amostras mais representativas que abrangem todo o espaço amostral. Métodos baseados em representatividade podem ser subdivididos em: (1) métodos baseados em densidade e (2) métodos baseados em diversidade. Métodos baseados em densidade envolvem selecionar amostras que melhor representam o conjunto de dados não rotulados. Um exemplo de método baseado em densidade é realizar o agrupamento (*clustering*) dos dados não rotulados e selecionar amostras próximas aos centroides. Por outro lado, métodos baseados em diversidade buscam selecionar amostras que se diferem das já rotuladas. Um exemplo de método baseado em diversidade é o (*Adaptive Batch Mode Active Learning – BMAL*) CARDOSO *et al.* (2017), que utiliza a divergência de Kullback-Leibler de classes de instâncias vizinhas.
- **Métodos baseados em influência:** tentam estimar o impacto no desempenho do modelo alvo. Pode ser subdividido em dois grupos de métodos: (1) métodos que tentam medir o impacto esperado e (2) métodos que se baseiam em outros métodos de aprendizado para estimar o impacto no modelo alvo, como aprendizado por reforço.

- **Métodos bayesianos:** tratam a rede sendo treinada como um modelo bayesiano. Normalmente, isso é feito deixando camadas de *dropout* ativas durante a inferência. Para cada dado, a previsão é feita diversas vezes. Com isso, o modelo dá confianças diferentes para o mesmo dado a cada vez que a inferência é realizada. O conjunto de inferências para um mesmo dado é então tratado como uma distribuição e os dados a serem rotulados são selecionados a partir da entropia dessa distribuição.
- **Métodos híbridos:** utilizam múltiplas estratégias para a seleção de dados, buscando um equilíbrio entre elas. As estratégias híbridas podem ser: (1) seriais, (2) seletivas ou (3) paralelas. Métodos híbridos seriais utilizam múltiplas estratégias sequencialmente, filtrando amostras pouco informativas. Métodos híbridos seletivos selecionam apenas uma estratégia para ser utilizada em uma rodada de aprendizado ativo, mas alteram possivelmente a estratégia utilizada em cada rodada. Por fim, métodos híbridos paralelos misturam múltiplos critérios a partir de funções de aquisição multi-objetivo.

2.1.1 Técnicas Clássicas de Aprendizado Ativo

Esta subseção cita as principais técnicas clássicas de Aprendizado Ativo. Essas técnicas consistem em definir uma pontuação para cada amostra dos dados não rotulados. As amostras com maior pontuação serão selecionadas para rotulação posteriormente.

- **Baixa confiança:** Esta estratégia consiste em selecionar amostras com baixa confiança de inferência. Para cada amostra, no conjunto de dados sem rótulo, realiza-se a inferência com o modelo atual e armazena-se a confiança c_n . A pontuação de uma amostra é dada por $s = 1 - c_n$. Essa é uma estratégia baseada em incerteza.
- **Entropia:** Esta estratégia consiste em selecionar amostras com maior entropia na probabilidade de classes. Para cada amostra não rotulada, realiza-se a inferência na amostra e aplica-se um *softmax* para obter uma distribuição de probabilidade \mathbf{X} . A entropia da amostra é calculada como $H_a = -\sum_{\mathbf{X}} p(x_i) \log_{10}(p(x_i))$. A pontuação da amostra é dada pela entropia da amostra normalizada pela soma das entropias de todas as amostras não rotuladas. Essa é uma estratégia baseada em incerteza.
- **Margem:** Esta estratégia consiste em selecionar as amostras com maior diferença entre a classe com maior confiança e a segunda classe com maior confiança. Para cada amostra, realiza a inferência com o modelo atual para cada amostra

no conjunto de dados sem rótulos e transforma a saída do modelo em uma distribuição de probabilidade (aplicando um *softmax*). As confianças da maior classe c_1 e da segunda maior classe c_2 são armazenadas. A pontuação da amostra é dada por $s = 1 - (c_1 - c_2)$. Essa é uma estratégia baseada em incerteza.

- **Similaridade de *embedding*:** Esta estratégia consiste selecionar as amostras com maior diferença nos *embeddings* (valores de uma camada intermediária da rede neural, neste caso, da penúltima camada). Para cada amostra dos dados já rotulados, realiza a inferência com o modelo atual sem a cabeça de detecção e armazena os *embeddings* em uma lista $L_{rotuladas} = \{l_1, l_2, \dots, l_n\}$. Para cada amostra dos dados não rotulados, realiza a inferência como anteriormente e armazena os *embeddings* l_a de cada amostra. Uma lista contendo a similaridade de cosseno entre o *embeddings* da amostra e os *embeddings* de todas as amostras já rotuladas é gerada para cada amostra $L_{sim_amostra} = \{\langle l_a, l_1 \rangle, \langle l_a, l_2 \rangle, \dots, \langle l_a, l_n \rangle\}$. A pontuação de uma amostra é dada por um menos o maior valor de similaridade de seu *logit* e o de todas as amostras já rotuladas $s = 1 - \max L_{sim_amostra}$. A similaridade de *embedding* é uma estratégia baseada em representatividade.

2.1.2 BALD

O Bayesian Active Learning by Disagreement (BALD) HOULSBY *et al.* (2011) é um método de aprendizado ativo para redes neurais profundas. Esse método é amplamente utilizado como linha de base pela literatura na experimentação com aprendizado ativo. Inicialmente, o BALD foi desenvolvido para utilização em modelos genéricos de aprendizado de máquina, mas sua relevância foi ampliada após uma adaptação do método para utilização em redes neurais profundas em tarefas de classificação de imagem GAL *et al.* (2017).

O objetivo do BALD é quantificar a incerteza de um modelo em dados de alta dimensionalidade, como imagens. Para isso, o método trata o modelo a ser otimizado como uma Rede Neural Bayesiana (*Bayesian Neural Network – BNN*) GAL e GHARAMANI (2016a), utilizando camadas de *dropout* de Monte Carlo (MC Dropout) GAL e GHARAMANI (2016b).

Redes neurais tradicionais são determinísticas, o resultado de uma inferência depende de seus pesos W e da entrada x . Por outro lado, em redes neurais Bayesianas, os pesos são tratados como uma distribuição de probabilidade $p(w)$. O treinamento de uma rede neural Bayesiana consiste em tratar os pesos atuais como uma *prior* e estimar uma *posterior* com base nos dados de treino D_{treino} , onde:

$$p(W|D_{treino}) = \frac{p(D_{treino}|W)p(W)}{p(D_{treino})}.$$

Como os parâmetros W agora possuem uma incerteza intrínseca, a predição do modelo para uma nova entrada x não é um único valor, mas sim a média de todas as predições possíveis ponderadas pela probabilidade dos pesos. Isso é formalizado pela marginalização sobre a posterior:

$$p(y = c|x, D_{treino}) = \int p(y = c|x, W)p(W|D_{treino})dW.$$

Na prática, resolver essa integral é computacionalmente inviável para redes profundas devido à alta dimensionalidade de W . Por isso, utiliza-se o MC Dropout. Ao manter as camadas de *dropout* ativadas durante a inferência, são realizadas múltiplas passagens estocásticas que funcionam como amostras da distribuição $p(W|D_{treino})$. A integral é então aproximada pela média dessas predições, permitindo estimar a incerteza necessária para o BALD.

O aprendizado ativo feito pelo BALD seleciona os dados para rotulação que maximizem a informação mútua entre as predições do modelo e seus parâmetros. A equação de pontuação de um dado é dada por:

$$\mathbb{I}[y, \omega | x, \mathcal{D}_{treino}] = \underbrace{\mathbb{H}[y | x, \mathcal{D}_{treino}]}_{\text{Incerteza Total}} - \underbrace{\mathbb{E}_{p(\omega|\mathcal{D}_{treino})}[\mathbb{H}[y | x, \omega]]}_{\text{Incerteza Aleatória}}.$$

O primeiro termo (Incerteza Total) mensura a confusão global do sistema ao calcular a entropia da média das predições, acumulando tanto a dúvida sobre qual é a classe correta quanto o ruído inerente aos dados. O segundo termo (Incerteza Aleatória), por sua vez, estima especificamente esse ruído ao avaliar a média de quão insegura foi cada inferência individualmente. Ao subtrair a estimativa de ruído (segundo termo) da incerteza total (primeiro termo), o método cancela a influência de dados ambíguos ou de má qualidade, resultando em uma pontuação que isola a Incerteza Epistêmica, ou seja, casos em que a informação é nítida, mas o modelo diverge internamente por ainda não ter aprendido os parâmetros necessários para aquela classificação.

2.1.3 Coreset

Até então, as estratégias mencionadas definem uma função de pontuação para um único dado. A principal desvantagem dessa abordagem é a agnosticidade ao conjunto selecionado. Considere que um modelo de classificação de imagem esteja sendo treinado em um *dataset* com imagens redundantes e dados não rotulados estão sendo selecionados para rotulação utilizando similaridade de *embedding*. Imagine que haja

n cópias de um dado com alta pontuação atribuída pela estratégia (ou seja, com baixa similaridade aos dados existentes no conjunto de dados rotulados). Se um *batch* de dados não rotulados for selecionado a partir dos k dados com maior pontuação atribuída pela estratégia, esse dado redundante com pontuação particularmente elevada será selecionado múltiplas vezes no mesmo *batch*. Em vista dessa “cegueira” aos dados já selecionados, as estratégias Coreset e BADGE (apresentadas na próxima subseção) propõem métodos sensíveis ao *batch*, atribuindo não uma pontuação, mas selecionando todo o subconjunto de dados a serem rotulados.

O Coreset SENER e SAVARESE (2018) é uma estratégia baseada em representatividade, onde o objetivo é selecionar o subconjunto de dados não rotulados que melhor represente o *dataset*. Para isso, o Coreset tenta resolver um problema de k -center nos *embeddings*, segundo a equação:

$$\min_{s \subseteq \mathcal{Z}_{pool}, |s|=k} \max_{i \in \mathcal{Z}_{pool}} \min_{j \in s} \Delta(\mathbf{z}_i, \mathbf{z}_j),$$

onde:

- \mathcal{Z}_{pool} é o conjunto do vetor de *embeddings* de todos os dados não rotulados.
- \mathbf{z}_i é o vetor de *embeddings* do dado i não rotulado.
- s é o subconjunto de tamanho equivalente ao tamanho do *batch* de dados a ser rotulado.
- \mathbf{z}_j é uma amostra já rotulada.
- $\Delta(\mathbf{z}_i, \mathbf{z}_j)$ representa a distância euclidiana entre as amostras.

Resolver essa equação é um problema NP-difícil, e sua solução é aproximada pelo algoritmo k -center guloso GONZALEZ (1985).

2.1.4 BADGE

O BADGE ASH *et al.* (2020) é uma estratégia que combina seleção por incerteza e diversidade. A premissa do BADGE é usar funções de influência KOH e LIANG (2017) para estimar o impacto de uma imagem não rotulada no modelo.

O algoritmo utilizado pelo BAGDE é reproduzido pelo Algoritmo 1. Primeiramente, o impacto de cada amostra é estimado calculando-se o gradiente da função de perda em relação aos pesos da última camada linear da rede. Isso é feito assumindo que a classe mais provável predita pela rede neural como ela está, é a correta, e simulando o treinamento dos dados não rotulados como se esse fosse seu rótulo. Em seguida, de posse do conjunto de vetores de todos os gradientes estimados dos

dados não rotulados, é realizado um agrupamento pelo algoritmo de *seeding* do K-means++ ARTHUR e VASSILVITSKII para seleção de amostras ao mesmo tempo diversas e com gradientes de grande magnitude.

Algoritmo 1: BADGE: Aprendizado Ativo em Lote via Embeddings de Gradiente Diversos.

Requer: Rede neural $f(x; \theta)$,
 Conjunto de exemplos não rotulados \mathcal{U} ,
 Número inicial de exemplos M ,
 Número de iterações T ,
 Número de exemplos em um lote B .

- 1 Conjunto de dados rotulados $\mathcal{S} \leftarrow M$ exemplos amostrados uniformemente ao acaso de \mathcal{U} juntamente com seus rótulos consultados.;
- 2 Treinar um modelo inicial θ_1 em \mathcal{S} minimizando $\mathbb{E}_{\mathcal{S}}[\ell_{CE}(f(x; \theta), y)]$.;
- 3 **for** $t = 1, 2, \dots, T$ **do**
- 4 **foreach** *exemplo* $x \in \mathcal{U} \setminus \mathcal{S}$ **do**
- 5 Calcular seu rótulo hipotético $\hat{y}(x) = h_{\theta_t}(x)$.;
- 6 Calcular o embedding de gradiente $g_x = \frac{\partial}{\partial \theta_{out}} \ell_{CE}(f(x; \theta), \hat{y}(x))|_{\theta=\theta_t}$, onde θ_{out} refere-se aos parâmetros da camada final (saída).;
- 7 **end**
- 8 Calcular \mathcal{S}_t , um subconjunto de $\mathcal{U} \setminus \mathcal{S}$, usando o algoritmo de inicialização k-MEANS++ em $\{g_x : x \in \mathcal{U} \setminus \mathcal{S}\}$ e consultar seus rótulos.;
- 9 $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_t$.;
- 10 Treinar um modelo θ_{t+1} em \mathcal{S} minimizando $\mathbb{E}_{\mathcal{S}}[\ell_{CE}(f(x; \theta), y)]$.;
- 11 **end**

Retorna: Modelo final θ_{T+1} .

Capítulo 3

Hybrid Strategy K-Center

3.1 Proposta

A estratégia proposta, *Hybrid Strategy K-Center – HSKC*, busca combinar diversas heurísticas a fim de explorar as vantagens de cada uma e, ao mesmo tempo, mitigar seus vieses. A proposta é gerar pontuações obtidas por heurísticas clássicas, que atuam em dados únicos e não em *batches*, e projetá-las em um espaço vetorial. Em seguida, a ideia é aplicar uma seleção baseada em diversidade geométrica no espaço vetorial criado. Essa estratégia tem a vantagem de ser sensível ao conjunto selecionado, ainda que as estratégias utilizadas para gerar o espaço vetorial não sejam.

Sendo D_u o conjunto de dados não rotulados. Para cada amostra $x_i \in D_u$, um vetor de características $v_i \in \mathbb{R}^M$ é calculado, onde M é o número de estratégias utilizadas. Para garantir robustez contra *outliers* durante a construção do espaço, aplica-se uma normalização Min-Max baseada nos quantis de 1% e 99%:

$$v'_{i,j} = \frac{\text{clamp}(v_{i,j}, q_{0.01,j}, q_{0.99,j}) - q_{0.01,j}}{(q_{0.99,j} - q_{0.01,j}) + \epsilon},$$

onde a função *clamp* limita os valores de $v_{i,j}$ ao intervalo $[q_{0.01,j}, q_{0.99,j}]$, (1º e o 99º percentil) arredondando valores fora da faixa dos percentis e ϵ evita divisões por zero.

Uma vez em posse do conjunto de características, k amostras são selecionadas a partir do algoritmo K-Centers guloso, assim como feito na estratégia Coreset.

O primeiro centro é selecionado como o de maior norma L2 no espaço de características, representando a amostra mais informativa segundo a agregação das estratégias. Para cada iteração, seleciona-se uma nova amostra u_i que maximiza a distância mínima ao conjunto de centros selecionados anteriormente S :

$$u^* = \arg \max_{u \in D_u \setminus S} \left(\min_{s \in S} \|v_u - v_s\|_2 \right).$$

3.1.1 Poda Bilateral

Diferente de abordagens que consideram todo o pool, como o Coreset e o BAGDE, o HSKC pode aplicar um filtro de informatividade combinada antes da seleção. Define-se a norma $\|v'_i\|_2$ como a pontuação agregada da amostra. O conjunto é então filtrado, mantendo apenas amostras cujas normas estejam entre os percentis definidos pelos parâmetros de *pruning*:

$$D_{filtro} = \{x_i \in D_u \mid \text{perc}_{low} \leq \|v'_i\|_2 \leq \text{perc}_{high}\}. \quad (3.1)$$

Essa técnica permite remover tanto dados redundantes ou muito “fáceis” (baixa norma) quanto possíveis ruídos ou *outliers* extremos (alta norma). Na implementação, o percentil inferior e superior a ser cortado pode ser ajustado por um hiperparâmetro.

3.2 Influência

Adicionalmente às estratégias selecionadas, o HSKC adiciona a influência da amostra ao vetor de pontuações. A influência é a estimativa da alteração nos pesos de um modelo ao ser treinado com uma amostra x , essa estimativa é aproximada por:

$$I(x) = \|z(x)\|_2 \cdot (1 - p_{\max}(x)),$$

onde $z(x)$ é o vetor de *embeddings* do modelo atual e p_{\max} é a probabilidade da classe predita com maior confiança. Enquanto as outras estratégias geram pontuações entre 0 e 1, a influência não possui um limite superior, por isso ela é normalizada antes de ser adicionada ao vetor de pontuações. A normalização é feita por:

$$I_{\text{final}}(x) = \log(1 + I(x)). \quad (3.2)$$

Utilizar a influência como pontuação é opcional na implementação, e pode ser desligada com uma *flag*.

3.3 Sobre o Consumo de Memória

Uma vantagem do HSKC sobre as outras estratégias sensíveis ao *batch* apresentadas é sua escalabilidade com relação ao uso de memória. Considere que:

- N é o tamanho do conjunto de dados não rotulado,
- D é a dimensão da camada de *embedding*,
- C é o número de classes do *dataset*,
- M é o número de heurísticas utilizadas no HSKC.

BADGE: O BADGE calcula o gradiente da função de perda em relação aos pesos da última camada da rede para cada amostra do pool. Como a última camada é uma matriz de pesos de tamanho $D \cdot C$, o gradiente resultante para cada amostra também tem dimensão $D \cdot C$, resultando em $N \cdot D \cdot C$. Assim, o consumo de memória do BADGE escala com $\mathcal{O}(N \cdot D \cdot C)$.

Coreset O Coreset armazena diretamente os vetores de *embedding* da penúltima camada da rede neural para cada amostra. Portanto, durante sua execução, o consumo de memória escala linearmente com N e D . Assim, o consumo de memória do Coreset escala com $(\mathcal{O}(N \cdot D))$.

HSKC: O HSKC armazena apenas um vetor de tamanho M para cada amostra. Como M tipicamente é pequeno (nos experimentos, $M = 4$: Entropia, BALD, Similaridade e Influência), o consumo de memória do HSKC permanece baixo, mesmo em *datasets* com muitas amostras. Assim, o consumo de memória do HSKC escala com $\mathcal{O}(N \cdot M)$.

Tabela 3.1: Estimativa de Memória para Seleção ($N = 50k$, $D = 1024$, $C = 100$).

Estratégia	Complexidade	Dependência	Estimativa de Memória*
BADGE	$\mathcal{O}(N \cdot D \cdot C)$	Classes e Modelo	≈ 19.1 GB
Coreset	$\mathcal{O}(N \cdot D)$	Modelo	≈ 195 MB
HSKC	$\mathcal{O}(N \cdot M)$	Apenas Amostras	≈ 0.8 MB

*Estimativa considerando precisão float32 (4 bytes).

A Tabela 3.1 ilustra a diferença de ordem de grandeza do consumo de memória de cada estratégia. Vale ressaltar que valores estimados são referente ao HSKC, não considerando possíveis diferentes estratégias que possam ser utilizadas em conjunto para a criação do espaço vetorial.

Capítulo 4

ALExp: Um *Framework* de Aprendizado Ativo

4.1 Motivação

A fim de realizar experimentos com aprendizado ativo, foi criado o ALExp, um *framework* de experimentação. Essa ferramenta tem como objetivo facilitar a aplicação de métodos de aprendizado ativo em modelos e *datasets* arbitrários. A ferramenta permite a criação de estratégias arbitrárias e é agnóstica ao *dataset*. O ALExp implementa todas as técnicas de aprendizado ativo apresentadas no Capítulo 2, além de uma nova técnica, o *Hybrid Strategy K-Center – HSKC*, que será apresentada no Capítulo 3. Esta seção foca nas ferramentas implementadas para a realização de experimentos de aprendizado ativo, detalhando o funcionamento do ALExp.

A Figura 4.1 ilustra o funcionamento do ALExp, indicando como uma estratégia pode ser utilizada para obtenção de pontuações. A figura também detalha o módulo em que cada componente é implementado.

4.1.1 Estratégias

O módulo `strats` foi criado para a implementação de estratégias de aprendizado ativo. O módulo conta com uma classe abstrata `Strategy`, com um método abstrato `get_scores`, que recebe um `Datapool` (contendo dados não rotulados), e, opcionalmente, um modelo do tipo `torch.nn.Module` e um `Dataset`, contendo dados rotulados.

Todas as estratégias aceitam um argumento inteiro `subset_size`. Quando um valor é passado para esse argumento, a estratégia é aplicada apenas em uma amostra aleatória da pool de tamanho `subset_size`; isso é útil para evitar que uma estratégia computacionalmente custosa seja aplicada no *datapool* completo em todos os passos de aprendizado ativo. Os argumentos opcionais do método `get_scores` podem ser

ExperimentDataset (para experimentos) e Datapool (para cenários práticos) são implementados no módulo datasets

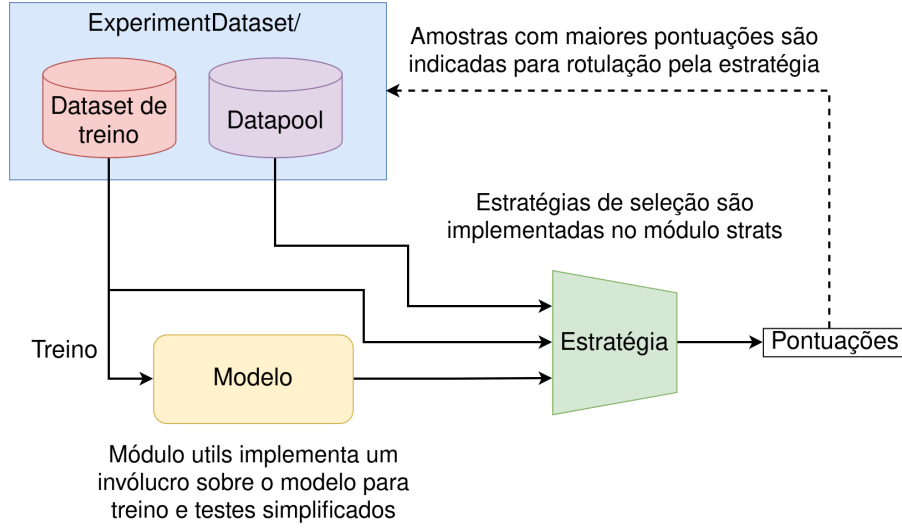


Figura 4.1: Visão geral do funcionamento do ALExp.

utilizados por estratégias específicas que dependem do estado do modelo atual e dos dados já rotulados. O método `get_scores` retorna um tensor contendo uma pontuação para cada dado não rotulado presente no `Datapool`. Essa pontuação é utilizada posteriormente na seleção de dados para rotulação.

A classe `Strategy` também conta com o método `__call__`, que orquestra a aplicação do método `get_scores`. Formalmente, seja $D_u = \{x_1, x_2, \dots, x_N\}$ o conjunto de dados total com N amostras, K o tamanho do subconjunto (`subset_size`) e $s(\cdot)$ a função de pontuação implementada em `get_scores`. O método define o conjunto de índices ativos \mathcal{I} para avaliação:

$$\mathcal{I} = \begin{cases} \{1, \dots, N\}, & \text{se } K \text{ é nulo ou } K \geq N \\ S \subset \{1, \dots, N\} \text{ tal que } |S| = K, & \text{caso contrário,} \end{cases}$$

onde o segundo caso representa uma amostragem uniforme sem reposição. O vetor de pontuação final $\mathbf{s} \in \mathbb{R}^N$ retornado pelo método é definido elemento a elemento como:

$$s_i = \begin{cases} s(x_i) & \text{se } i \in \mathcal{I} \\ -\infty & \text{se } i \notin \mathcal{I}. \end{cases}$$

A atribuição de $-\infty$ atua como uma máscara, garantindo que amostras não avaliadas não sejam selecionadas em operações subsequentes de maximização (como `argmax` ou `top-k`).

A implementação de estratégias de aprendizado ativo é feita a partir da especificação da classe `Strategy`. Uma estratégia específica é implementada criando

uma nova classe que herda da classe `Strategy` e implementa o método `get_scores` específico para a estratégia. Por padrão, foram implementadas as estratégias `EmbeddingSimilarity`, `LowConfidence`, `Entropy` e `Margin`, `BALD`, `BADGE`, `Coreset` e `HSKC` – uma estratégia proposta no Capítulo 3 – como detalhadas na Seção 2.1.1, além da `RandomChoice`, que sorteia aleatoriamente as pontuações.

A Figura 4.2 retrata o diagrama UML das classes implementadas no módulo `strats`. Indicando a classe `Strategy` implementa uma abstração, enquanto detalhes da função de seleção específicas das estratégias são implementadas pelas classes derivadas.

4.1.2 *Datasets*

O módulo `Datasets` foi criado a fim de abstrair os detalhes de implementação de *datasets* de diferentes tipos para a realização de aprendizado ativo. O objetivo deste módulo é permitir que o *dataset* do Pytorch possa ser utilizado com as estratégias implementadas no módulo `strats`. Este módulo também implementa classes que permitem a realização de experimentos de aprendizado ativo utilizando um *dataset* já rotulado, removendo os rótulos e adicionando conforme necessário.

O principal objetivo do módulo `Datasets` é manter o controle dos dados rotulados e não rotulados em experimentos de aprendizado ativo. A estrutura do módulo baseia-se no gerenciamento eficiente de índices, evitando a duplicação de dados em memória. Para isso, foi implementada a classe auxiliar `Indexer`, que atua como um *wrapper* sobre o *dataset* original. A função desta classe é criar uma visualização virtual do conjunto de dados, mapeando um subconjunto de índices ativos para os dados reais. Além de gerenciar o acesso aos dados, a `Indexer` é responsável por aplicar transformações específicas (como aumento de dados ou normalização) no momento do acesso, garantindo que o *pipeline* de processamento seja consistente.

A classe `DataPool` herda de `Dataset` e especializa-se no gerenciamento das amostras não rotuladas. Ela encapsula os dados disponíveis para rotulação e implementa métodos otimizados para a manipulação de tensores de índices, como `remove_data` e `pop_data`. A classe implementa o `get_top_n`, que serve como interface para as estratégias de consulta: ele recebe um vetor de pontuações e retorna o subconjunto das N amostras mais informativas, prontas para serem movidas para o conjunto de treino.

No nível mais alto de abstração encontra-se a classe `ExperimentDataset`, que orquestra todo o ciclo de vida do experimento. Essa classe mantém o estado global da simulação, dividindo o *dataset* original em duas instâncias de `Indexer`:

- `self.dataset`: Representa o conjunto de treinamento atual, contendo apenas as amostras cujos rótulos foram “revelados”.

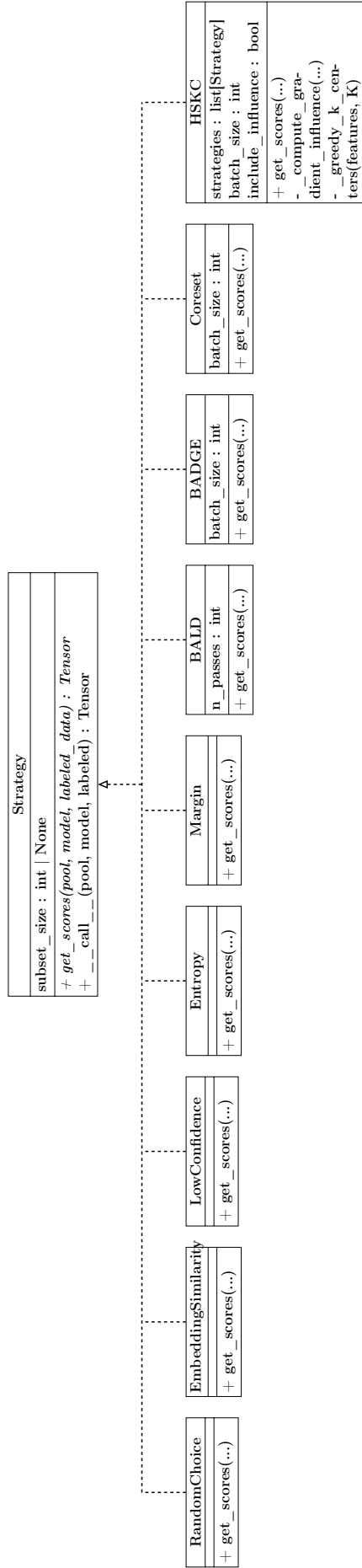


Figura 4.2: Implementação das estratégias no *framework*.

- `self.pool`: Representa o conjunto de candidatos, contendo as amostras ainda não rotuladas disponíveis para seleção.

A classe `ExperimentDataset` fornece a API principal para a execução dos laços de aprendizado ativo. O método `random_label_n` é utilizado para a inicialização do experimento (*cold start*), selecionando aleatoriamente amostras do *pool* e transferindo-as para o conjunto de treino. Já o método `label_top_n` concretiza o passo de aprendizado ativo: ele recebe as pontuações calculadas por uma estratégia, identifica as amostras prioritárias na *pool*, remove-as da lista de não rotulados e as adiciona ao conjunto de treinamento, simulando a ação de um oráculo. Adicionalmente, o método `sample_n` permite a amostragem aleatória simples do *pool*.

A Figura 4.3 representa o diagrama UML da implementação das classes `Indexer`, `DataPool` e `ExperimentDataset`. No diagrama UML, a classe `Dataset` mais acima refere-se de forma simplificada à implementação da classe `Dataset` da biblioteca Pytorch.

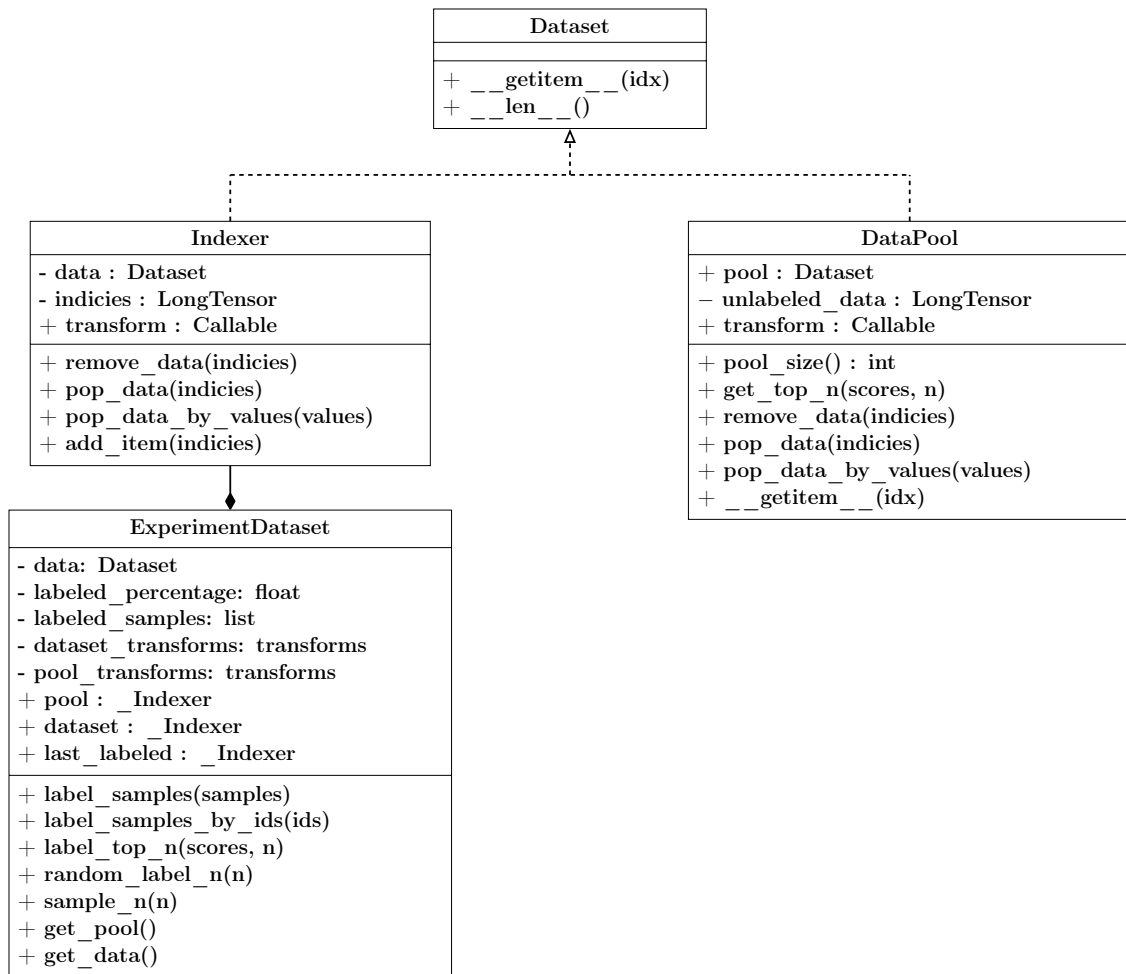


Figura 4.3: Diagrama UML das classes implementadas no arquivo `datasets`.

4.1.3 Utilidades

A biblioteca implementa a classe `ModelWrapper`, que funciona como um invólucro sobre um modelo e implementa laços de treino e teste. A classe pode receber uma função de otimização e de *scheduler*, e gerencia sua aplicação. A classe `ModelWrapper` conta com os métodos `set_optimizer`, que recebe um otimizador e seus argumentos, `set_scheduler`, que recebe um *scheduler* e seus argumentos e `reset`, que *reseta* um modelo para seus pesos originais, útil para experimentação em aprendizado ativo. O otimizador e o *scheduler* também são *resetados*. O `ModelWrapper` também implementa os métodos `train`, responsável por treinar o modelo por um número de épocas, o método `train` também implementa um *early-stopping* opcional, com um limiar de *loss* no *dataset* de validação. O método `train_one_epoch` é responsável por implementar uma época de treinamento. O método `test` recebe um *dataset* de teste e retorna a acurácia e a perda de teste.

4.2 Laço de aprendizado ativo com a ferramenta

A Figura 4.4 ilustra como a ferramenta pode ser utilizada para experimentação com aprendizado ativo. Primeiramente, um *dataset* rotulado é dividido entre os dados utilizados no experimento e os dados utilizados no teste do modelo final. Os dados utilizados no experimento são subdivididos em *dataset* de treino e *datapool*. O *dataset* de treino simula a parcela de dados já rotulados, esses dados, juntamente com seus rótulos, podem ser acessados como uma instância da classe `Dataset` do Pytorch a partir do método `get_data` do `ExperimentDataset`. Já os dados no *datapool* têm seus rótulos ocultos, e os dados (sem os rótulos) podem ser acessados como uma instância da classe `Dataset` a partir do método `get_pool`. A quantidade de dados iniciais já rotulados em um experimento pode ser passada como um argumento na criação de uma instância da classe `ExperimentDataset`. Também é possível definir, não uma porcentagem de dados rotulados inicialmente, mas uma lista contendo os índices dos dados enviados ao *dataset* de treino.

O laço de um experimento consiste primeiramente em treinar um modelo com os dados presentes no *dataset* de treino de um `ExperimentDataset`. Após o treino, é testado se a quantidade de dados já rotulados no *dataset* de treino é maior que um *budget*. Este *budget* é definido a priori, antes do início do experimento. Caso o *budget* seja atingido, o modelo é testado com o *dataset* de treino, separado anteriormente, e o experimento chega ao fim. Caso contrário, uma estratégia escolhida pelo usuário anteriormente é executada. Essa estratégia pode depender do modelo atual, dos dados presentes no *datapool* e dos dados já rotulados presentes no *dataset* de treino. A estratégia retorna uma lista de pontuações, sendo que essa lista tem

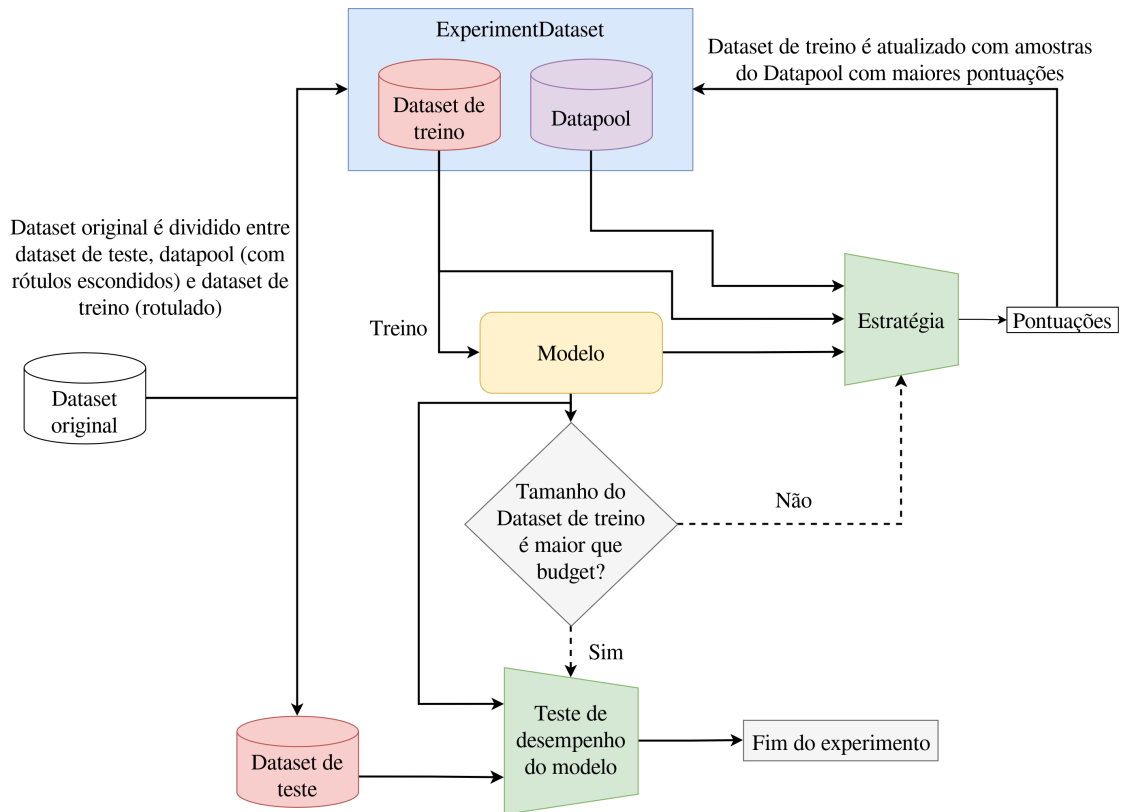


Figura 4.4: Laço de aprendizado ativo de um experimento com o arcabouço desenvolvido.

o mesmo tamanho da quantidade de dados presentes no *datapool*, e atribui uma pontuação para cada dado presente no *datapool*. O método `label_top_n` recebe a lista de pontuação retornada pela estratégia e seleciona os n dados com maiores pontuações. Esses dados são retirados do *datapool* e enviados para o *dataset* de treino com seus respectivos rótulos. Com isso, uma nova rodada é iniciada e o modelo pode ser treinado com os novos dados rotulados, até que o *budget* seja atingido.

Capítulo 5

Experimentos em Aprendizado Ativo

5.1 Procedimentos Experimentais

Os experimentos abaixo comparam as diferentes estratégias presentes no *framework* desenvolvido. O experimento consiste primeiramente em selecionar um *dataset* rotulado, utilizar `ExperimetDataset` para simular um ambiente de aprendizado ativo, separando o *dataset* em uma parte com rótulos e uma parte sem rótulos. Uma parcela do *dataset* pré-estabelecida é selecionada aleatoriamente para simular os dados rotulados iniciais. O modelo é treinado nos dados rotulados iniciais e testado nos dados de teste. Após o treino inicial, a estratégia atual sendo testada é aplicada e pontua os dados não rotulados, os n dados com maior pontuação são retirados do conjunto de dados sem rótulos e adicionados para o conjunto de dados com rótulos, e o modelo é treinado novamente. Isso é repetido até que o conjunto de dados com rótulos atinja um tamanho estabelecido.

O experimento é repetido para cada uma das estratégias implementadas (`RandomChoice`, `EmbeddingSimilarity`, `LowConfidence`, `Entropy`, `Margin`, `BALD`, `Coreset`, `BADGE` e `HCK`). Todos os experimentos foram realizados com o modelo *Resnet18*. Vale ressaltar que, para o funcionamento da estratégia `BALD`, foi necessário adicionar uma camada de *dropout* antes da camada de *output*. A priori, apenas o experimento com `BALD` utilizou o modelo com camada de *dropout*. No entanto, essa abordagem conferiu uma vantagem desleal em favor do `BALD`, uma vez que o modelo com *dropout* desempenhou melhor logo na primeira rodada de treino, antes de qualquer rodada de aprendizado ativo. Portanto, todos os experimentos foram realizados com a *Resnet18* com camadas de *dropout*.

O Algoritmo 2 ilustra em pseudo-código o procedimento experimental. Em todos os experimentos, o `HCK` utilizou as estratégias `BALD`, `Entropy`, `EmbeddingSimilarity` e a flag `include_influence` estava marcada como `True`. Os hiperparâmetros `pruning_low` e `pruning_high` foram mantidos em 0,2 e 1 (sem poda superior) nos

datasets SVHN e MNIST e em todos os experimentos 0,7 e 1 no CIFAR10 e CIFAR10 redundante. O `subset_size` de todas as estratégias foi mantido em 20000.

Algoritmo 2: Procedimento experimental.

Input: Conjunto de Treino \mathcal{D}_{train} , Conjunto de Teste \mathcal{D}_{test} , Orçamento de Query b , Ciclos C

Output: Lista de Acurácias A

```

1  $A \leftarrow []$ ;
2  $N_{rotulado} \leftarrow \text{TamanhoInicial}$ ;
3  $\mathcal{D}_{rotulado}, \mathcal{D}_{pool} \leftarrow \text{DivisãoInicial}(\mathcal{D}_{train}, N_{rotulado})$ ;
4  $\mathcal{M} \leftarrow \text{InicializarModelo}(\text{ResNet18}, \text{SGD})$ ;
5  $\mathcal{L}_{train} \leftarrow \text{CriarDataLoader}(\mathcal{D}_{rotulado})$ ;
6  $\mathcal{M} \leftarrow \text{Treinar}(\mathcal{M}, \mathcal{L}_{train}, \text{epochs} = 50)$ ;
7  $acc \leftarrow \text{Avaliar}(\mathcal{M}, \mathcal{D}_{test})$ ;
8  $A \leftarrow A \cup \{acc\}$ ;
9 for  $k \leftarrow 1$  to  $C$  do
10    $scores \leftarrow \text{Estrategia}(\mathcal{D}_{pool}, \mathcal{M}, \mathcal{D}_{rotulado})$ ;
11    $amostras \leftarrow \text{SelecionarTopN}(scores, b)$ ;
12    $\mathcal{D}_{rotulado} \leftarrow \mathcal{D}_{rotulado} \cup amostras$ ;
13    $\mathcal{D}_{pool} \leftarrow \mathcal{D}_{pool} \setminus amostras$ ;
14    $\mathcal{M} \leftarrow \text{ResetarPesos}(\mathcal{M})$ ;
15    $\mathcal{L}_{train} \leftarrow \text{CriarDataLoader}(\mathcal{D}_{rotulado})$ ;
16    $\mathcal{M} \leftarrow \text{Treinar}(\mathcal{M}, \mathcal{L}_{train}, \text{epochs} = 50)$ ;
17    $acc \leftarrow \text{Avaliar}(\mathcal{M}, \mathcal{D}_{test})$ ;
18    $A \leftarrow A \cup \{acc\}$ ;
19 end
20 return  $A$ ;

```

Para a realização dos experimentos, foram utilizados os *datasets* CIFAR10 KRIZHEVSKY (2009), SVHN NETZER *et al.* (2011) e MNIST DENG (2012). Esses *datasets* têm classes identicamente distribuídas. Adicionalmente, também foi gerada uma versão redundante do *CIFAR10*, com distribuição de classes não identicamente distribuídas e dados de pouca variabilidade na mesma classe. O processo de geração desse *dataset* está detalhado no Seção 5.2. O Código 5.1 especifica como foi adicionada a camada de *dropout* à Resnet18 utilizando as bibliotecas Pytorch e Torchvision.

```

1 class ResNet18Dropout(ResNet):
2     def __init__(self, num_classes=10):
3         super(ResNet18Dropout, self).__init__(BasicBlock, [2,
4         2, 2], num_classes=num_classes)
5         self.fc = nn.Sequential(
6             nn.Dropout(p=0.5),
7             nn.Linear(512 * BasicBlock.expansion, num_classes)

```

Código 5.1: Implementação da ResNet18 com Dropout, utilizando Pytorch e a Resnet do Torchvision.

Todos os experimentos foram realizados utilizando o otimizador *Stochastic Gradient Descent – SGD* e o *scheduler Cosine Annealing*. A taxa de aprendizado (*learning rate*) e o número de épocas por rodada variou conforme o experimento, de acordo com a quantidade de dados rotulados disponível. Não foi utilizado *early-stopping*.

Foram realizadas *transforms* nos dados, estas *transforms* precisaram ser separadas em *transforms* aplicadas no *datapool* e *transforms* aplicadas no *dataset* rotulado. As *transforms* aplicadas no *datapool* são voltadas a adequar os dados ao formato esperado pela *ResNet*. Por outro lado, as *transforms* aplicadas ao *dataset* rotulado envolvem tanto adequar os dados ao formato esperado pela *ResNet* quanto aumentações. Essa divisão é necessária, pois não faz sentido realizar aumento em dados não rotulados, uma vez que não estão sendo utilizados para treino. A Tabela 5.1 detalha as transformações aplicadas aos *datasets*.

5.2 Geração do *Datasets* Redundantes

Para os experimentos de aprendizado ativo, foi criado um subconjunto desbalanceado e redundante do *dataset* CIFAR10. Para isso, um algoritmo gera a distribuição de classes de cauda longa, com decaimento exponencial, seguindo a fórmula $n_c = n_0 * k^c$, onde n_c é a c -ésima classe com mais dados e k é um parâmetro de decaimento. Ademais, em uma mesma classe foi definida uma redundância r fixa. Por exemplo, se uma classe possui 1000 amostras e $r = 5$, então apenas 200 amostras desta classe foram obtidas do CIFAR10 original, as demais foram obtidas artificialmente a partir da aumento dessas 200 originais.

As aumentações utilizadas foram um recorte aleatório com preenchimento de 4 pixels e um espelhamento horizontal com probabilidade de 0.5.

Um *dataset* gerado com fator de redundância $r = 5$, o fator de decaimento foi escolhido de forma que a classe mais abundante tenha 10 vezes mais dados que a menos redundante. Esse *dataset* foi salvo em disco para a realização dos experimentos. A Figura 5.1 mostra a quantidade de amostras por classe, e a Tabela 5.3 relaciona a quantidade de amostras reais e geradas artificialmente.

Um procedimento similar foi feito com o *dataset* MNIST, gerando a distribuição de classes representada pela Figura 5.2

Tabela 5.1: Transformações aplicadas aos *Datasets*.

<i>Dataset</i>	Transforms do Dataset Rotulado	Transforms do Datapool
CIFAR10 / CIFAR10 Red.	ToImage() RandomCrop(32, padding=4) RandomHorizontalFlip(p=0.5) ToDtype(float32, scale=True) Normalize($\mu = (0.4914, 0.4822, 0.4465),$ $\sigma = (0.2023, 0.1994, 0.2010))$	ToImage() ToDtype(float32, scale=True) Normalize($\mu = (0.4914, 0.4822, 0.4465),$ $\sigma = (0.2023, 0.1994, 0.2010))$
MNIST	Grayscale(num_output_channels=3) Resize((32, 32)) RandomRotation(10) RandomCrop(32, padding=4) ToImage() ToDtype(float32, scale=True) Normalize($\mu = (0.1307, 0.1307, 0.1307),$ $\sigma = (0.3081, 0.3081, 0.3081))$	Grayscale(num_output_channels=3) Resize((32, 32)) ToImage() ToDtype(float32, scale=True) Normalize($\mu = (0.1307, 0.1307, 0.1307),$ $\sigma = (0.3081, 0.3081, 0.3081))$
SVHN	ToTensor() Normalize($\mu = (0.4377, 0.4438, 0.4728),$ $\sigma = (0.1980, 0.2010, 0.1970))$	ToTensor() Normalize($\mu = (0.4377, 0.4438, 0.4728),$ $\sigma = (0.1980, 0.2010, 0.1970))$

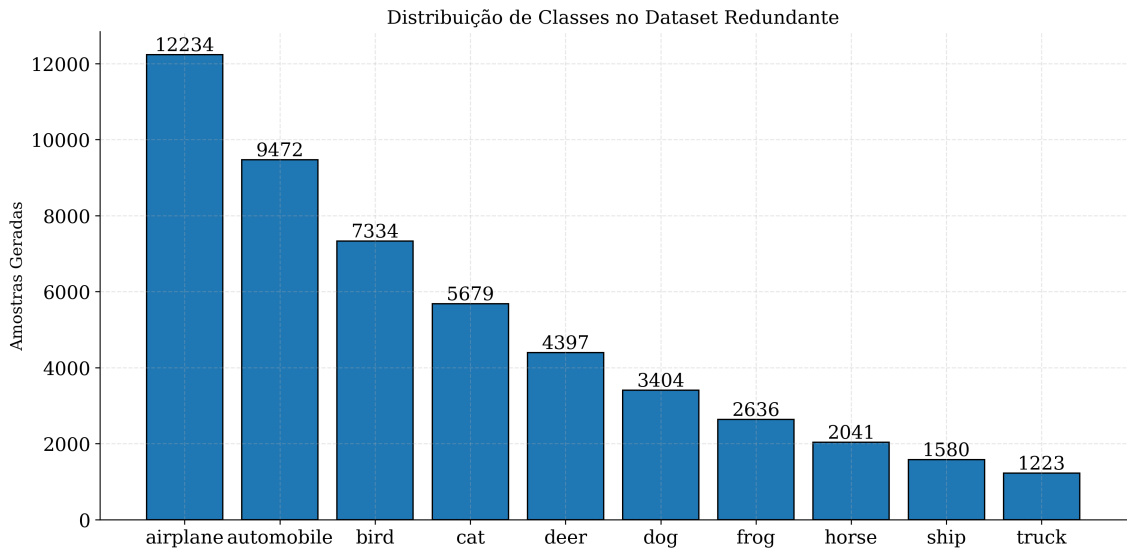


Figura 5.1: Distribuição de classes no *dataset* CIFAR-10 redundante gerado, com fator de redundância 5.

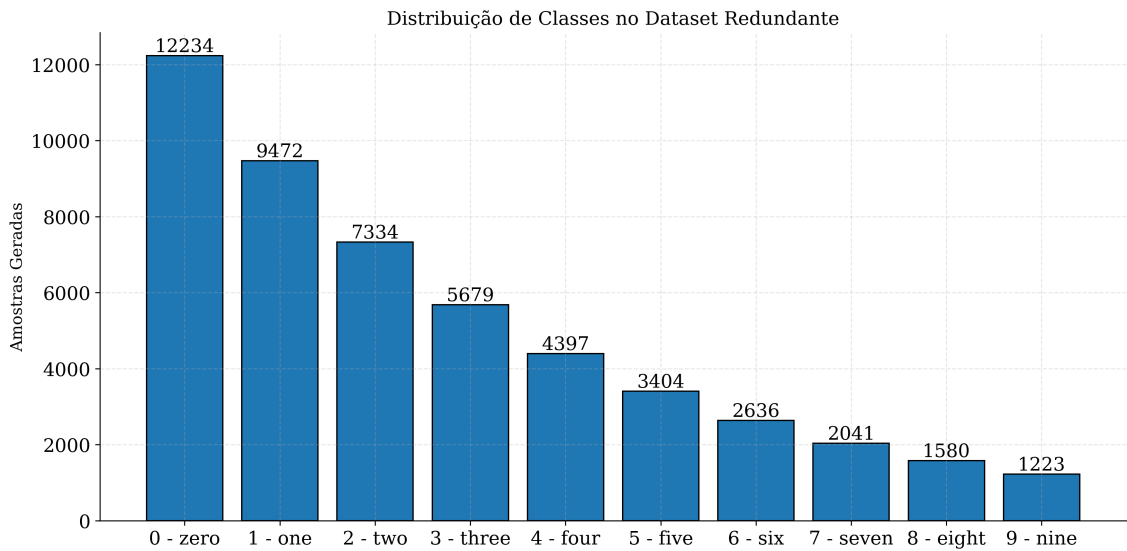


Figura 5.2: Distribuição de classes no *dataset* MNIST redundante gerado, com fator de redundância 5.

Classe	Total Gerado	Originais Usadas
airplane	12.234	2.447
automobile	9.472	1.895
bird	7.334	1.467
cat	5.679	1.136
deer	4.397	880
dog	3.404	681
frog	2.636	528
horse	2.041	409
ship	1.580	316
truck	1.223	245
Total	50.000	10.006

Tabela 5.2: Distribuição de amostras e originais utilizadas por classe no *dataset* gerado.

Classe	Total Gerado	Originais Usadas
0 - zero	12.234	2.447
1 - one	9.472	1.895
2 - two	7.334	1.467
3 - three	5.679	1.136
4 - four	4.397	880
5 - five	3.404	681
6 - six	2.636	528
7 - seven	2.041	409
8 - eight	1.580	316
9 - nine	1.223	245
Total	50.000	10.004

Tabela 5.3: Distribuição de amostras e originais utilizadas por classe no *dataset* gerado.

5.3 Experimentos

5.3.1 Experimentos em pequena escala com *datasets* clássicos

Os experimentos realizados no SVHN e no MNIST começaram com 256 amostras iniciais e 64 amostras adicionais selecionadas a cada ciclo pela estratégia. O número de ciclos foi 6, totalizando 640 amostras. Ambos foram treinados por 50 épocas com *learning rate* de 0,01. A Figura 5.3 ilustra a evolução do modelo com relação ao número de dados rotulados por cada estratégia. A Tabela 5.4 mostra a média e o desvio padrão obtidos para o SVHN. A Figura 5.4 mostra a evolução das estratégias no MNIST, e a Tabela 5.5 detalha os resultados.

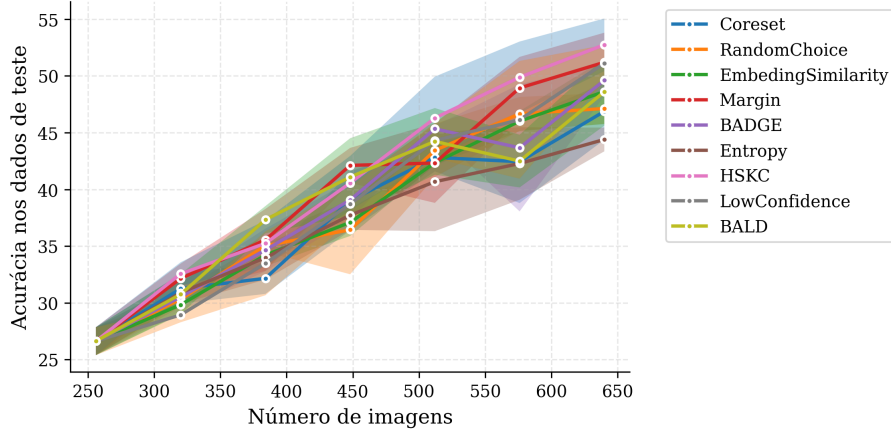


Figura 5.3: Teste de estratégias no SVHN.

Tabela 5.4: Comparação da Acurácia de Teste (%) ao Longo dos Ciclos de Aprendizado Ativo no *dataset* SHVN.

Estratégia	Ciclo 0 (256)	Ciclo 1 (320)	Ciclo 2 (384)	Ciclo 3 (448)	Ciclo 4 (512)	Ciclo 5 (576)	Ciclo 6 (640)
Coreset	26.63 ± 1.23	31.26 ± 1.31	32.17 ± 1.37	39.00 ± 2.60	42.82 ± 1.22	42.44 ± 3.64	46.88 ± 2.50
RandomChoice	26.63 ± 1.23	30.32 ± 0.66	35.04 ± 0.53	36.44 ± 3.92	43.44 ± 2.34	46.67 ± 1.48	47.13 ± 1.35
EmbeddingSim	26.63 ± 1.23	29.84 ± 0.78	34.21 ± 0.98	37.09 ± 1.19	42.36 ± 0.22	46.02 ± 0.88	48.62 ± 2.86
Margin	26.63 ± 1.23	32.15 ± 1.32	35.55 ± 2.72	42.12 ± 1.54	42.33 ± 3.51	48.94 ± 2.74	51.20 ± 2.61
BADGE	26.63 ± 1.23	30.53 ± 1.23	34.67 ± 1.87	39.07 ± 1.70	45.34 ± 0.26	43.67 ± 5.60	49.62 ± 1.36
Entropy	26.63 ± 1.23	30.90 ± 0.98	33.99 ± 1.93	37.73 ± 1.30	40.68 ± 4.36	42.28 ± 3.19	44.40 ± 1.04
HSKC	26.63 ± 1.23	32.57 ± 1.02	35.25 ± 2.02	40.57 ± 2.38	46.29 ± 3.65	49.88 ± 3.15	52.75 ± 2.31
LowConfidence	26.63 ± 1.23	28.95 ± 0.65	33.49 ± 2.82	38.72 ± 1.52	44.31 ± 1.55	46.14 ± 5.18	51.12 ± 1.61
BALD	26.63 ± 1.23	30.76 ± 1.79	37.33 ± 1.20	41.09 ± 3.42	44.23 ± 2.94	42.50 ± 2.32	48.61 ± 2.98

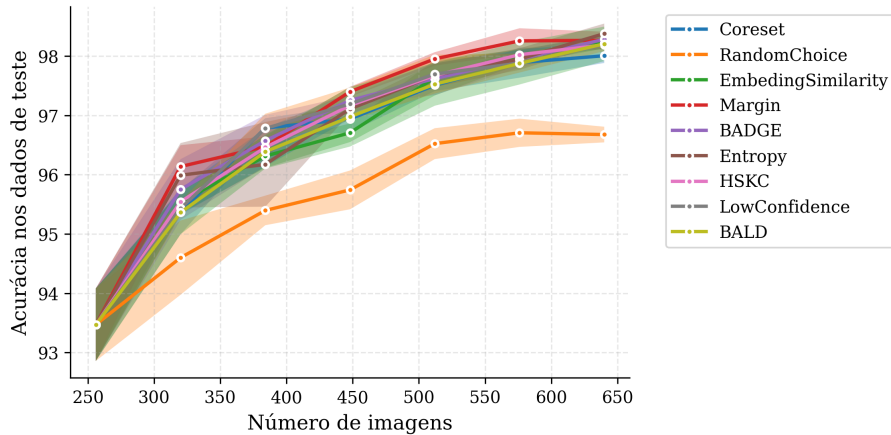


Figura 5.4: Teste de estratégias no MNIST.

O experimento realizado com SVHN demonstrou um cenário inicial desafiador, com o modelo sendo iniciado com apenas 26% de acurácia nos dados de teste. Nesse cenário, onde os *embeddings* do modelo ainda variam muito, nota-se uma vantagem para as técnicas baseadas em incerteza (como *Margin* e *LowConfidence*) e abordagens híbridas (como o *HSKC*). Essas estratégias conseguiram guiar o aprendizado

Tabela 5.5: Comparação da Acurácia de Teste (%) ao Longo dos Ciclos de Aprendizado Ativo no *dataset* MNIST.

Estratégia	Ciclo 0 (256)	Ciclo 1 (320)	Ciclo 2 (384)	Ciclo 3 (448)	Ciclo 4 (512)	Ciclo 5 (576)	Ciclo 6 (640)
Coreset	93.47 ± 0.61	95.37 ± 0.37	96.78 ± 0.16	96.94 ± 0.22	97.51 ± 0.09	97.89 ± 0.25	98.00 ± 0.12
RandomChoice	93.47 ± 0.61	94.60 ± 0.63	95.40 ± 0.25	95.74 ± 0.33	96.52 ± 0.26	96.71 ± 0.24	96.68 ± 0.13
EmbeddingSim	93.47 ± 0.61	95.54 ± 0.19	96.31 ± 0.20	96.71 ± 0.16	97.64 ± 0.30	97.99 ± 0.12	98.24 ± 0.11
Margin	93.47 ± 0.61	96.14 ± 0.36	96.47 ± 0.17	97.40 ± 0.08	97.95 ± 0.11	98.26 ± 0.21	98.27 ± 0.15
BADGE	93.47 ± 0.61	95.75 ± 0.51	96.57 ± 0.44	97.25 ± 0.08	97.62 ± 0.28	97.95 ± 0.14	98.28 ± 0.10
Entropy	93.47 ± 0.61	95.99 ± 0.54	96.17 ± 0.71	97.11 ± 0.08	97.68 ± 0.20	97.95 ± 0.18	98.38 ± 0.17
HSKC	93.47 ± 0.61	95.54 ± 0.22	96.46 ± 0.32	97.17 ± 0.23	97.67 ± 0.20	98.02 ± 0.08	98.19 ± 0.07
LowConfidence	93.47 ± 0.61	95.43 ± 0.12	96.78 ± 0.25	97.19 ± 0.28	97.70 ± 0.33	97.88 ± 0.16	98.19 ± 0.11
BALD	93.47 ± 0.61	95.36 ± 0.37	96.39 ± 0.28	96.98 ± 0.50	97.53 ± 0.36	97.88 ± 0.36	98.21 ± 0.29

de forma mais eficiente, encerrando o sexto ciclo com acurácias médias superiores a 51%, um ganho considerável em relação à amostragem aleatória (`RandomChoice`).

Por outro lado, os resultados obtidos no MNIST iniciam com acurácias em torno de 93% no primeiro ciclo. Nesse cenário, a maioria do dados não rotulados passa a ser composta por exemplos simples que a rede já domina, fazendo com que a escolha aleatória desperdice o reduzido orçamento de anotação sorteando amostras que não agregam novo conhecimento. Em contraste, a pequena margem de erro do modelo concentra-se em casos atípicos e ambíguos. Nesse contexto, estratégias de aprendizado ativo focadas em incerteza (como `Entropy`, `Margin`, `LowConfidence` e `BALD`) identificam precisamente as amostras que confundem as fronteiras de decisão, enquanto abordagens focadas em diversidade ou métodos híbridos (como `Coreset`, `BADGE` e `HSKC`) garantem a exploração de diferentes tipos de falhas no espaço de características. Dessa forma, a superioridade dessas estratégias no MNIST decorre não da quantidade de dados adicionada a cada ciclo, mas da maximização do ganho informacional obtido por cada nova amostra.

5.3.2 Experimentos com CIFAR-10 e CIFAR-10 redundante

Foram realizados experimentos de maior escala com o CIFAR-10 e o CIFAR-10 redundante. Esses experimentos começaram com 5120 amostras iniciais e 1024 amostras adicionais selecionadas a cada ciclo pela estratégia. O número de ciclos foi cinco, totalizando 10240 amostras. Nos experimentos com CIFAR-10 os modelos foram treinados por 150 épocas com taxa de aprendizado de 0,01. A Figura 5.5 ilustra a evolução do modelo com relação ao número de dados rotulados por cada estratégia no CIFAR-10. A Tabela 5.6 mostra a média e o desvio padrão obtidos para o CIFAR-10. A Figura 5.6 e a Tabela 5.7 fazem o mesmo para a versão redundante do CIFAR-10. Também foi executado um experimento em menor escala com o CIFAR-10, com 512 amostras rotuladas iniciais e incrementos de 128 amostras por ciclo, com 50 épocas por rodada de treino. A Figura 5.7 ilustra a evolução do modelo enquanto a Tabela 5.8 detalha o resultado do experimento. Por fim, foi realizado um teste de maior escala com as estratégias `Coreset`, `BADGE` e `HSKC` no

CIFAR-10 redundante. Esse teste teve início em 25600 amostras com incrementos de 35840 por ciclo. A rede foi treinada com 200 épocas e taxa de aprendizado de 0.01. Nesse experimento o argumento `pruning_low` do HSKC foi mantido em 0.5.

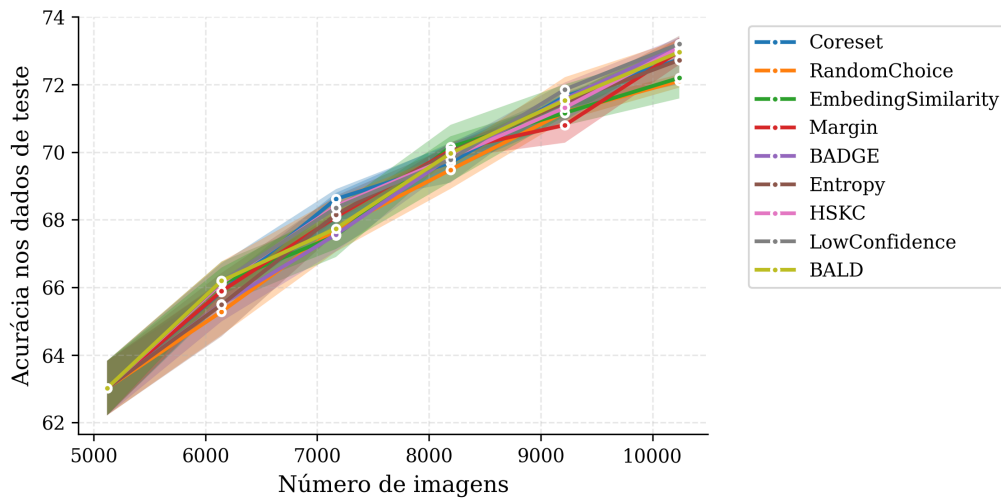


Figura 5.5: Teste de estratégias no CIFAR10.

Tabela 5.6: Comparação da Acurácia de Teste (%) no CIFAR-10 ao Longo dos Ciclos de Aprendizado Ativo.

Estratégia	Ciclo 0 (5120)	Ciclo 1 (6144)	Ciclo 2 (7168)	Ciclo 3 (8192)	Ciclo 4 (9216)	Ciclo 5 (10240)
Coreset	63.02 ± 0.81	65.87 ± 0.28	68.61 ± 0.17	69.69 ± 0.69	71.53 ± 0.11	72.74 ± 0.07
RandomChoice	63.02 ± 0.81	65.29 ± 0.70	67.73 ± 0.67	69.48 ± 0.56	71.19 ± 0.24	72.11 ± 0.20
EmbeddingSim	63.02 ± 0.81	66.13 ± 0.64	67.53 ± 0.63	70.16 ± 0.31	71.16 ± 0.37	72.20 ± 0.61
Margin	63.02 ± 0.81	65.89 ± 0.30	68.09 ± 0.10	70.06 ± 0.23	70.80 ± 0.52	72.99 ± 0.40
BADGE	63.02 ± 0.81	65.51 ± 0.51	67.57 ± 0.50	69.87 ± 0.34	71.58 ± 0.48	73.14 ± 0.25
Entropy	63.02 ± 0.81	65.49 ± 0.94	68.16 ± 0.56	70.01 ± 0.16	71.47 ± 0.16	72.73 ± 0.72
HSKC	63.02 ± 0.81	66.13 ± 0.59	68.40 ± 0.50	69.85 ± 0.38	71.32 ± 0.39	73.09 ± 0.14
LowConfidence	63.02 ± 0.81	66.17 ± 0.59	68.36 ± 0.30	69.78 ± 0.20	71.86 ± 0.36	73.21 ± 0.12
BALD	63.02 ± 0.81	66.20 ± 0.39	67.74 ± 0.61	69.97 ± 0.84	71.54 ± 0.47	72.97 ± 0.07

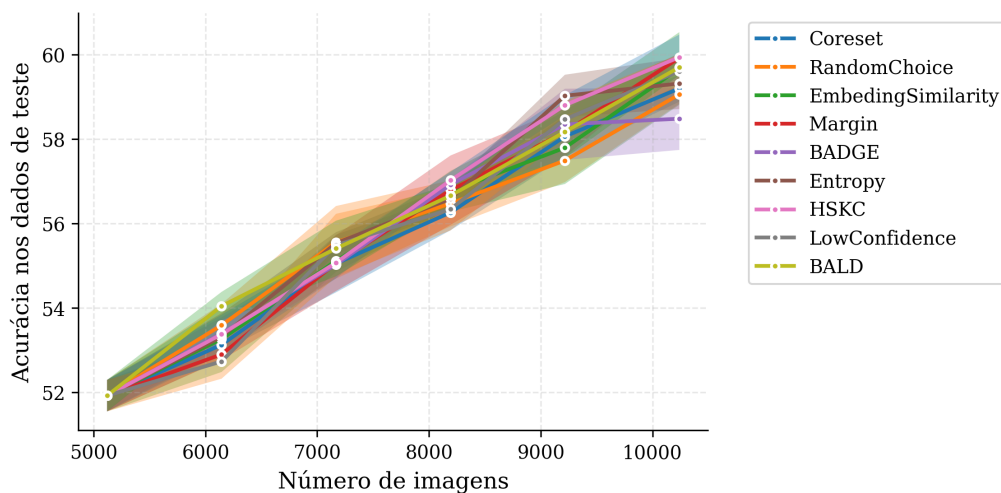


Figura 5.6: Teste de estratégias no CIFAR10 redundante.

Tabela 5.7: Comparação da Acurácia de Teste (%) no CIFAR-10 Redundante ao Longo dos Ciclos de Aprendizado Ativo.

Estratégia	Ciclo 0 (5120)	Ciclo 1 (6144)	Ciclo 2 (7168)	Ciclo 3 (8192)	Ciclo 4 (9216)	Ciclo 5 (10240)
Coreset	51.92 ± 0.37	53.12 ± 0.15	55.68 ± 0.19	56.26 ± 0.42	58.06 ± 0.48	59.19 ± 0.31
RandomChoice	51.92 ± 0.37	53.59 ± 0.51	55.55 ± 0.86	56.47 ± 0.54	57.49 ± 0.51	59.06 ± 0.22
EmbeddingSim	51.92 ± 0.37	53.26 ± 0.78	55.11 ± 0.26	56.76 ± 0.48	57.79 ± 0.86	59.67 ± 0.86
Margin	51.92 ± 0.37	52.90 ± 0.13	55.06 ± 0.66	56.79 ± 0.82	58.16 ± 0.58	59.92 ± 0.11
BADGE	51.92 ± 0.37	53.39 ± 0.38	55.03 ± 0.30	56.92 ± 0.13	58.35 ± 0.84	58.48 ± 0.74
Entropy	51.92 ± 0.37	53.32 ± 0.56	55.53 ± 0.26	56.58 ± 0.30	59.02 ± 0.50	59.31 ± 0.60
HSKC	51.92 ± 0.37	53.38 ± 0.46	55.07 ± 0.37	57.02 ± 0.20	58.80 ± 0.24	59.94 ± 0.53
LowConfidence	51.92 ± 0.37	52.73 ± 0.41	55.48 ± 0.74	56.34 ± 0.50	58.47 ± 0.44	59.61 ± 0.40
BALD	51.92 ± 0.37	54.04 ± 0.34	55.41 ± 0.66	56.66 ± 0.27	58.17 ± 0.62	59.70 ± 0.18

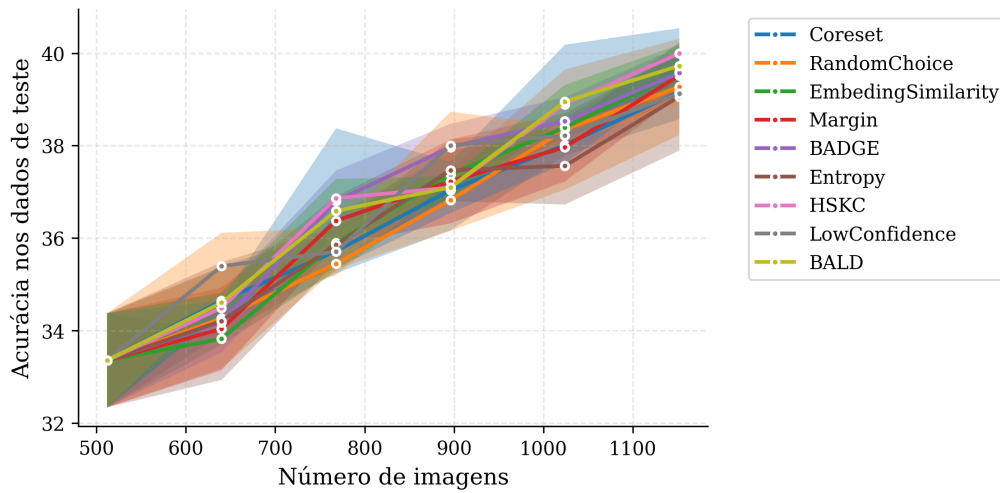


Figura 5.7: Teste de estratégias no CIFAR-10 Redundante, com número de amostras iniciais reduzidas.

Tabela 5.8: Comparação da Acurácia de Teste (%) no CIFAR-10 Redundante em Escala Reduzida ao Longo dos Ciclos de Aprendizado Ativo.

Estratégia	Ciclo 0 (512)	Ciclo 1 (640)	Ciclo 2 (768)	Ciclo 3 (896)	Ciclo 4 (1024)	Ciclo 5 (1152)
Coreset	33.36 ± 1.02	34.64 ± 0.74	35.72 ± 0.48	37.04 ± 0.87	38.01 ± 0.23	39.17 ± 0.59
RandomChoice	33.36 ± 1.02	34.27 ± 1.08	35.45 ± 0.12	36.82 ± 0.67	38.35 ± 1.30	39.27 ± 1.04
EmbeddingSimilarity	33.36 ± 1.02	33.82 ± 0.12	35.90 ± 0.60	37.38 ± 0.47	38.39 ± 0.21	39.59 ± 0.45
Margin	33.36 ± 1.02	34.04 ± 0.89	36.38 ± 0.53	37.23 ± 0.91	37.97 ± 0.74	39.52 ± 0.44
BADGE	33.36 ± 1.02	34.18 ± 0.64	36.80 ± 0.66	37.97 ± 0.50	38.53 ± 0.51	39.57 ± 0.56
Entropy	33.36 ± 1.02	34.21 ± 1.27	35.86 ± 0.38	37.47 ± 0.67	37.56 ± 0.84	39.06 ± 1.17
HSKC	33.36 ± 1.02	34.49 ± 0.16	36.86 ± 1.51	37.09 ± 0.55	38.90 ± 1.29	39.99 ± 0.55
LowConfidence	33.36 ± 1.02	35.39 ± 0.72	35.71 ± 0.52	38.01 ± 0.73	38.22 ± 0.19	39.13 ± 0.21
BALD	33.36 ± 1.02	34.60 ± 0.22	36.59 ± 0.69	37.10 ± 0.24	38.95 ± 0.36	39.73 ± 0.46

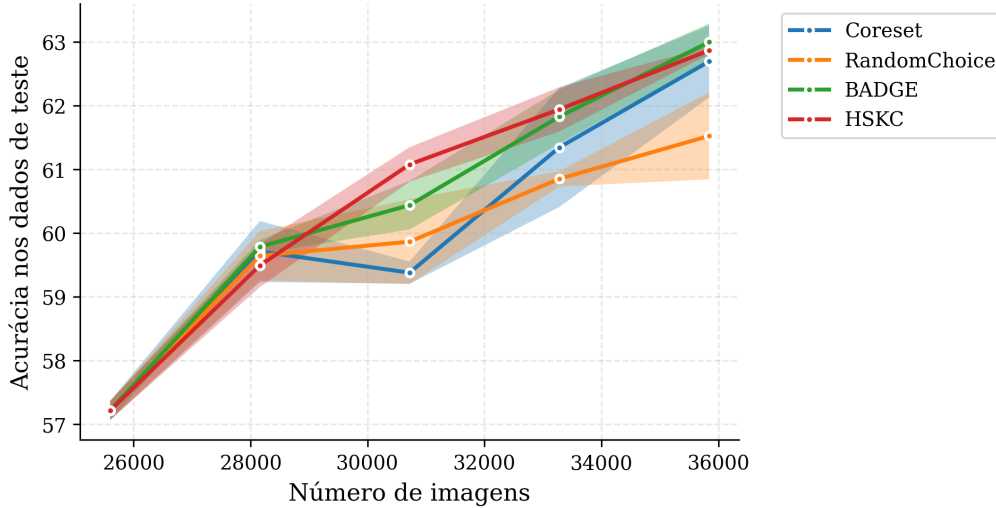


Figura 5.8: Teste de estratégias no CIFAR10 redundante em grande escala.

Tabela 5.9: Comparação da Acurácia de Teste (%) no CIFAR-10 Redundante em Grande Escala ao Longo dos Ciclos de Aprendizagem Ativo.

Estratégia	Ciclo 0 (25600)	Ciclo 1 (28160)	Ciclo 2 (30720)	Ciclo 3 (33280)	Ciclo 4 (35840)
Coreset	57.21 ± 0.15	59.71 ± 0.48	59.38 ± 0.17	61.34 ± 0.93	62.69 ± 0.56
RandomChoice	57.21 ± 0.15	59.64 ± 0.39	59.86 ± 0.66	60.85 ± 0.12	61.52 ± 0.68
BADGE	57.21 ± 0.15	59.78 ± 0.11	60.44 ± 0.38	61.83 ± 0.42	63.00 ± 0.29
HSKC	57.21 ± 0.15	59.49 ± 0.34	61.08 ± 0.27	61.94 ± 0.35	62.87 ± 0.06

O CIFAR-10 é um *dataset* desafiador para Aprendizagem Ativo, em sua versão, o *dataset* tem uma distribuição de classes uniforme, com dados variados. Nesse cenário, a maioria das estratégias obteve uma pequena margem em relação à seleção aleatória, porém o ganho obtido pela estratégia é ofuscado pelo ganho obtido por simplesmente adicionar mais dados rotulados.

Os experimentos com a versão redundante do CIFAR-10 apresentam um comportamento semelhante, mas com a adição de redundância, a margem de acurácia obtida sobre a seleção aleatória torna-se um pouco mais perceptível.

5.3.3 Experimentos com MNIST redundante

Por fim, foram com o *dataset* MNIST redundante. O experimento com menor escala, representado pela Figura 5.10 e pela Tabela 5.11 iniciou com 256 amostras rotuladas e 128 amostras adicionais selecionadas a cada ciclo pela estratégia. O número de ciclos foi 6, totalizando 1024 amostras. A taxa de aprendizado foi mantida em 0,01 e o número de épocas foi ajustado para 50, em vista da quantidade de dados. Já a Figura 5.9 e a Tabela 5.10 representam um experimento com 5120 amostras rotuladas inicialmente e incrementos de 1028 por ciclo. O modelo foi treinado por 100 épocas por episódios.

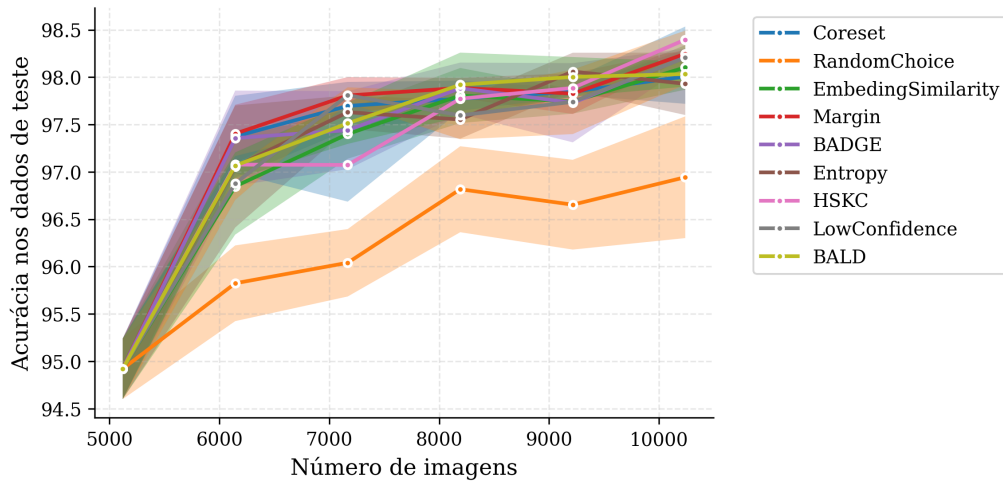


Figura 5.9: Teste de estratégias no MNIST redundante.

Tabela 5.10: Comparação da Acurácia de Teste (%) no MNIST Redundante ao Longo dos Ciclos de Aprendizagem Ativo.

Estratégia	Ciclo 0 (5120)	Ciclo 1 (6144)	Ciclo 2 (7168)	Ciclo 3 (8192)	Ciclo 4 (9216)	Ciclo 5 (10240)
Coreset	94.92 ± 0.32	97.37 ± 0.43	97.70 ± 0.25	97.77 ± 0.19	97.85 ± 0.03	98.01 ± 0.29
RandomChoice	94.92 ± 0.32	95.82 ± 0.40	96.04 ± 0.36	96.82 ± 0.45	96.65 ± 0.47	96.94 ± 0.64
EmbeddingSimilarity	94.92 ± 0.32	96.85 ± 0.51	97.40 ± 0.26	97.81 ± 0.29	97.74 ± 0.13	98.10 ± 0.23
Margin	94.92 ± 0.32	97.40 ± 0.31	97.81 ± 0.19	97.89 ± 0.11	97.83 ± 0.22	98.25 ± 0.04
BADGE	94.92 ± 0.32	97.36 ± 0.50	97.44 ± 0.41	97.88 ± 0.27	97.73 ± 0.42	98.21 ± 0.09
Entropy	94.92 ± 0.32	97.06 ± 0.64	97.63 ± 0.18	97.55 ± 0.20	98.06 ± 0.20	97.93 ± 0.33
HSKC	94.92 ± 0.32	97.08 ± 0.09	97.07 ± 0.39	97.77 ± 0.13	97.89 ± 0.10	98.40 ± 0.14
LowConfidence	94.92 ± 0.32	96.88 ± 0.17	97.81 ± 0.08	97.60 ± 0.25	97.74 ± 0.34	98.21 ± 0.29
BALD	94.92 ± 0.32	97.07 ± 0.14	97.51 ± 0.22	97.92 ± 0.34	98.00 ± 0.21	98.03 ± 0.13

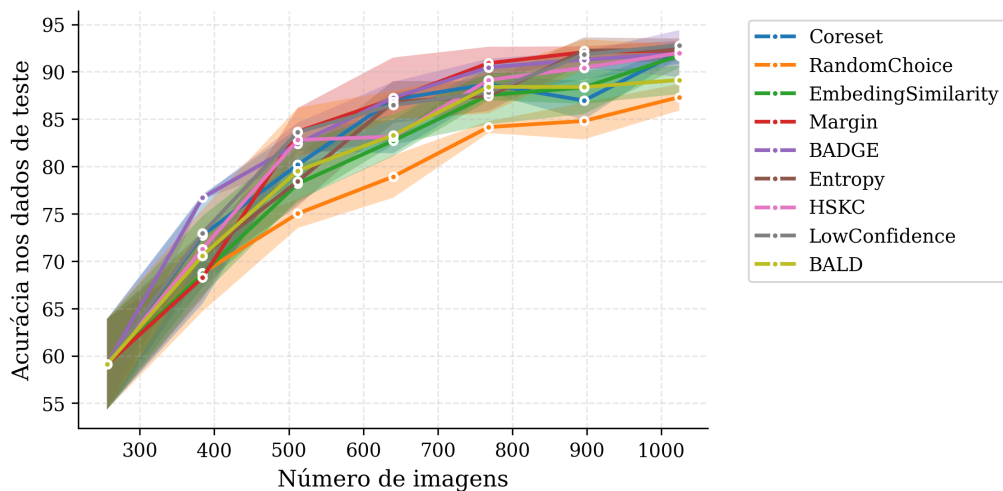


Figura 5.10: Teste de estratégias no MNIST redundante, com número de amostras iniciais reduzidas.

Esses experimentos evidenciam a importância de realizar Aprendizagem Ativa para seleção de novas amostras em um cenário com amostras saturadas. O experimento

Tabela 5.11: Comparação da Acurácia de Teste (%) no MNIST Redundante em Escala Eeduzida ao Longo dos Ciclos de Aprendizado Ativo.

Estratégia	Ciclo 0 (256)	Ciclo 1 (320)	Ciclo 2 (384)	Ciclo 3 (448)	Ciclo 4 (512)	Ciclo 5 (576)	Ciclo 6 (640)
Coreset	59.12 ± 4.78	72.73 ± 0.35	80.18 ± 1.75	87.07 ± 0.72	88.69 ± 0.92	86.96 ± 2.04	92.08 ± 0.54
RandomChoice	59.12 ± 4.78	68.76 ± 4.08	75.03 ± 1.52	78.93 ± 2.23	84.16 ± 0.62	84.81 ± 1.95	87.32 ± 1.41
EmbeddingSimilarity	59.12 ± 4.78	68.69 ± 1.43	78.19 ± 2.00	82.72 ± 0.42	87.53 ± 3.09	88.34 ± 2.80	91.71 ± 1.25
Margin	59.12 ± 4.78	68.25 ± 2.18	83.67 ± 2.50	87.12 ± 4.38	90.91 ± 1.73	92.07 ± 0.61	92.27 ± 1.00
BADGE	59.12 ± 4.78	76.71 ± 0.16	82.40 ± 2.22	87.27 ± 1.67	90.46 ± 0.92	91.28 ± 0.86	91.93 ± 2.47
Entropy	59.12 ± 4.78	70.56 ± 1.78	78.43 ± 2.62	86.81 ± 2.21	87.44 ± 1.57	92.23 ± 1.40	92.33 ± 1.18
HSKC	59.12 ± 4.78	71.28 ± 5.78	82.81 ± 1.16	83.15 ± 1.76	89.13 ± 1.65	90.42 ± 1.26	91.99 ± 1.13
LowConfidence	59.12 ± 4.78	72.95 ± 2.51	83.63 ± 2.57	86.50 ± 1.50	87.81 ± 2.19	91.83 ± 1.59	92.80 ± 0.08
BALD	59.12 ± 4.78	70.56 ± 4.19	79.55 ± 3.07	83.30 ± 2.22	88.41 ± 1.42	88.38 ± 1.76	89.13 ± 1.48

de maior escala evidencia que rapidamente as estratégias encontram as amostras mais informacionais entre as não rotuladas, dando um salto inicial considerável logo nos primeiros ciclos.

Já no experimento de menor escala, apesar do ganho de desempenho das estratégias ser considerável, isso ocorre ao longo dos ciclos. Isso dá-se porque, nesse cenário com poucos dados rotulados inicialmente, aumentar a quantidade bruta de dados rotulados ainda fornece um ganho considerável de desempenho ao modelo.

5.4 Conclusões Sobre os Experimentos

Os experimentos exibem a eficácia dos métodos implementados no ALExp, dado que selecionar amostras aleatórias (não realizar Aprendizado Ativo) mostrou-se como a pior opção em diversos cenários. Enquanto técnicas clássicas mostraram-se eficazes em alguns cenários, em outros o viés adicionado pela técnica e a “cegueira” de *batch* fizeram com que algumas dessas técnicas desempenhassem pior que a escolha aleatória.

O HSKC mostrou-se uma escolha robusta quando comparado a outras estratégias sensíveis ao *batch*, como o BADGE e o Coreset, exibindo resultados equiparáveis ou melhores, principalmente em cenários com escassez inicial de amostras rotuladas. Isso ocorre pois o HSKC, por utilizar técnicas clássicas na criação do espaço vetorial, é menos dependente da qualidade dos *embeddings* atuais do modelo treinado do que o BADGE e o Coreset, enquanto mantém uma característica sensível ao *batch*, que contribui para a qualidade dessas estratégias em cenários maiores.

Capítulo 6

Conclusão

Este trabalho apresenta o ALExp, um *framework* para realização de experimentos e aplicação prática de aprendizado ativo. O *framework* conta com recursos para gerenciar o ciclo de vida de experimentos em aprendizado ativo de forma eficiente. A classe `ExperimentDataset` abstrai o processo de separação e monitoramento de dados rotulados e não rotulados, além de permitir a aplicação de aumentações diferentes para dados da *pool* e dados rotulados. O `ExperimentDataset` também conta com métodos fáceis de rotulação, integrado com a classe `Strategy`. Por sua vez, a classe `Strategy` provê uma abstração robusta às diferentes estratégias de aprendizado ativo, sendo versátil tanto a estratégias que estabelecem uma pontuação para cada dado quanto estratégias que selecionam os dados em si. Ademais, o *framework* implementa algumas das principais técnicas da literatura (`RandomChoice`, `EmbeddingSimilarity`, `LowConfidence`, `Entropy`, `Margin`, `BALD`, `Coreset`, `BADGE`).

Este trabalho também apresenta uma nova estratégia híbrida, o Hybrid Strategy K-Center (HSKC), que agrega diferentes estratégias originalmente não formuladas para aplicação em lotes de uma maneira sensível ao lote, enquanto desempenhou consistentemente melhor do que as heurísticas nos experimentos. Uma das vantagens do HSKC é sua escalabilidade com relação ao uso de memória, a estratégia consegue ser aplicada em conjuntos de dados maiores enquanto continua competitiva com relação ao Coreset e o BADGE nos experimentos.

Há planos de expandir o ALExp futuramente para abranger estratégias diversas de aprendizado ativo para a tarefa de detecção de objeto. Outro ponto a ser abordado futuramente é sobre os hiperparâmetros do HSKC. Apesar do HSKC se mostrar promissor, atualmente o usuário deve escolher quais estratégias serão utilizadas em conjunto e quais os valores dos hiperparâmetros `pruning_low` e `pruning_high`. Nesse sentido, há planos para realizar testes variando as estratégias e os hiperparâmetros para averiguar quais as combinações recomendadas a depender do caso de uso, além de experimentos de escalas ainda maiores.

Referências Bibliográficas

- KRIZHEVSKY, A. *Learning multiple layers of features from tiny images*. Relatório técnico, 2009.
- DENG, J., DONG, W., SOCHER, R., et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- VAN ENGELEN, J. E., HOOS, H. H. “A survey on semi-supervised learning”, *Machine Learning*, v. 109, n. 2, pp. 373–440, 2020. doi: 10.1007/s10994-019-05855-6.
- SENER, O., SAVARESE, S. “Active Learning for Convolutional Neural Networks: A Core-Set Approach”. In: *International Conference on Learning Representations*, 2018. Disponível em: <<https://openreview.net/forum?id=H1aIuk-RW>>.
- ASH, J. T., ZHANG, C., KRISHNAMURTHY, A., et al. “Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds”. 2020. Disponível em: <<https://arxiv.org/abs/1906.03671>>.
- ATIGHEHCHIAN, P., BRANCHAUD-CHARRON, F., FREYBERG, J., et al. “Baal, a bayesian active learning library”. <https://github.com/baal-org/baal/>, 2022.
- DANKA, T., HORVATH, P. “modAL: A modular active learning framework for Python”, Disponível em: <<https://github.com/modAL-python/modAL>>. available on arXiv at <https://arxiv.org/abs/1805.00979>.
- LI, D., WANG, Z., CHEN, Y., et al. “A Survey on Deep Active Learning: Recent Advances and New Frontiers”, *IEEE Transactions on Neural Networks and Learning Systems*, v. 36, n. 4, pp. 5879–5899, 2025. doi: 10.1109/TNNLS.2024.3396463.

- HOULSBY, N., HUSZÁR, F., GHAHRAMANI, Z., et al. “Bayesian Active Learning for Classification and Preference Learning”. 2011. Disponível em: <<https://arxiv.org/abs/1112.5745>>.
- DENG, L. “The mnist database of handwritten digit images for machine learning research”, *IEEE Signal Processing Magazine*, v. 29, n. 6, pp. 141–142, 2012.
- NETZER, Y., WANG, T., COATES, A., et al. “Reading Digits in Natural Images with Unsupervised Feature Learning”. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. Disponível em: <http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf>.
- FANG, M., LI, Y., COHN, T. “Learning How to Active Learn: A Deep Reinforcement Learning Approach”. ago. 2017.
- XIE, B., YUAN, L., LI, S., et al. “Active Learning for Domain Adaptation: An Energy-Based Approach”, *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 36, n. 8, pp. 8708–8716, jun. 2022. ISSN: 2374-3468. doi: 10.1609/aaai.v36i8.20850.
- WANG, D., SHANG, Y. “A New Active Labeling Method for Deep Learning”. In: *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 112–119, Beijing, China, jul. 2014. IEEE. doi: 10.1109/IJCNN.2014.6889457.
- CARDOSO, T. N., SILVA, R. M., CANUTO, S., et al. “Ranked batch-mode active learning”, *Information Sciences*, v. 379, pp. 313–337, 2017. ISSN: 0020-0255. doi: <https://doi.org/10.1016/j.ins.2016.10.037>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0020025516313949>>.
- GAL, Y., ISLAM, R., GHAHRAMANI, Z. “Deep Bayesian Active Learning with Image Data”. 2017. Disponível em: <<https://arxiv.org/abs/1703.02910>>.
- GAL, Y., GHAHRAMANI, Z. “Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference”. 2016a. Disponível em: <<https://arxiv.org/abs/1506.02158>>.
- GAL, Y., GHAHRAMANI, Z. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. 2016b. Disponível em: <<https://arxiv.org/abs/1506.02142>>.

GONZALEZ, T. F. “Clustering to minimize the maximum intercluster distance”, *Theoretical Computer Science*, v. 38, pp. 293–306, 1985. ISSN: 0304-3975. doi: [https://doi.org/10.1016/0304-3975\(85\)90224-5](https://doi.org/10.1016/0304-3975(85)90224-5). Disponível em: <<https://www.sciencedirect.com/science/article/pii/0304397585902245>>.

KOH, P. W., LIANG, P. “Understanding Black-box Predictions via Influence Functions”. In: *Proceedings of the 34th International Conference on Machine Learning*, pp. 1885–1894. PMLR, jul. 2017.

ARTHUR, D., VASSILVITSKII, S. “K-Means++: The Advantages of Careful Seeding”, .