Deep learning para classificaçao de conjuntos abertos

João Victor Mergner Scaravonatti
Ciência da computação
Instituto Federal Catarinense
Videira, Brasil
jvscaravonatti@gmail.com

***Abstract*—Neste trabalho será utilizado um método para classificação de conjuntos abertos utilizando *deep learning*. Em cenários de classificação/reconhecimento tradicionais, todas as classes são conhecidas a priori, ou seja, na etapa de treinamento. Já em cenários de conjuntos abertos, nem todas as classes são conhecidas nesta fase, portanto, classes desconhecidas podem surgir na fase de teste. Como solução para este problemas foram desenvolvidas diversas técnicas, no entanto, o método apresentado neste trabalho utilizará deep learning. Para abordar o problema em questão, o método proposto utiliza *intra-class splitting* para dividir o conjunto de dados em classes típicas e atípicas, sendo que o último conjunto será responsável por modelar as classes desconhecidas.**

***Palavras-chave*—Classificação de conjuntos abertos, deep learning, intra-class splitting**

I. Introdução

De fato, uma das tarefas mais desafiadoras do aprendizado de máquina é a classificação, uma vez que os modelos tradicionais possuem limitações quanto à classificação de amostras desconhecidas pelo modelo. No geral, para esta tarefa é necessário que os algoritmos sejam abastecidos com amostras de classes conhecidas (previamente anotadas por humanos) durante as fases de treinamento e de teste [1]. Desta maneira, o modelo será treinado para aprender a discriminar dentre as classes que foram apresentadas ao modelo de classificação. Tradicionalmente, esta abordagem é conhecida como classificação de *conjuntos fechados*, uma vez que todas as amostras já estão previamente *fechadas* em um contexto definido anteriormente por um especialista humano. Contudo, em cenários mais realístico, nem sempre as amostras disponíveis são previamente conhecidas, ou seja, não raro os problemas de classificação se apresentam como de *conjuntos abertos*, já que durante a fase de teste do modelo, além de amostras de classes conhecidas, também existirão amostras de classes que são desconhecidas pelo modelo [2].

O maior desafio da classificação dentro de conjuntos abertos é fazer com que os classificadores, além de classificarem as classes conhecidas, lidem com as amostras pertencentes as classes desconhecidas. Esse tipo de classificação se difere de um método de classificação tradicional, uma vez que é preciso classificar, além das amostras de classes conhecidas, as amostras desconhecidas como sendo de classes desconhecidas. Desta forma, não basta apenas classificar cada entrada com a classe que mais se assemelha a ela, é preciso definir se ela é conhecida ou não [2].

Recentemente, os modelo de aprendizado profundo (*deep learning*) tem alcançado o estado da arte na resolução de diversos problemas, em especial para aqueles problemas onde as informações que podem ser interpretadas por pessoas, como na detecção de objetos em imagens por exemplo. Este tipo de modelo computacional é uma extensão das Redes Neurais Artificiais (RNAs) e sua principal particularidade é possuir diversas camadas de neurônios ocultas entre as camadas de entrada e saída, por isto o termo "profundo". Além disso, os modelos de aprendizado profundo também possuem capacidade de aprender de maneira automatizada características e relacionamentos entre diversos tipos de dados complexos (como imagens, textos e áudios), o que os torna muito eficientes em problemas de classificação [3].

Recentemente, várias técnicas de aprendizado profundo conseguiram obter performances similares e até melhores que humanos em diferentes domínios. Entre elas estão as Redes Neurais Convolucionais (CNNs) (também conhecidas como *ConvNets*), que são inspiradas no córtex visual dos animais [3]. Além de todos os benefícios herdados das redes profundas, as CNNs são conhecidas por reduzir o número de parâmetros utilizados em uma RNA tradicional. Deste modo, essa característica de redução de parâmetros torna a CNN muito eficiente quando estão sendo tratados dados complexos e multidimensionais [4].

Problemas que envolvem a classificação de objetos em imagens se encaixam perfeitamente dentro do problema de classificação de conjuntos abertos. Isso se dá pelo fato de que, em muito problemas de classificação de imagens, é impossível ter conhecimento prévio de todas as possíveis classes existentes. Nesse sentido, é importante que um modelo computacional seja generalista o suficiente para distinguir dentre classes conhecidas das não conhecidas [2]. Neste ponto, o aprendizado profundo é naturalmente uma boa abordagem que pode ser utilizada para contornar o problema de classificação em problemas de conjuntos abertos aplicados à imagens. De acordo com 3, as CNNs apresentaram excelentes resultados nas áreas de visão computacional e processamento de imagens. Portanto, o problema que se pretende abordar neste trabalho é a classificação de conjuntos abertos aplicados à classificação de imagens e a principal hipótese de trabalho é que os métodos

de aprendizado profundo (*deep learning*), em especial as CNNs, podem ser interessantes ferramentas para viabilizar a modelagem computacional do problema abordado.

II. Referencial teórico

A. Aprendizado profundo

Aprendizado profundo (*Deep learning*) é uma das técnicas que revolucionaram a área de inteligência artificial nos últimos anos, resolvendo problemas que não foram solucionados durante vários anos pelos modelos tradicionais de aprendizado de máquina. Os modelos de aprendizado profundo são variantes das redes neurais artificiais e se diferem pela característica de possuírem muitas camadas de neurônios [3].

Essa técnica é conhecida pelo seu desempenho na análise de dados complexos, multidimensionais e também em dados com ruído, isto é, se apresentam fora do padrão existente nos outros dados do conjunto. A capacidade dos modelos de aprendizado para o aprendizado de características hierárquicas em diversos tipos de dados os tornam muito eficientes para tarefas de reconhecimento de padrões, regressão e problemas de classificação semi-supervisionados e não-supervisionados [3].

B. Redes neurais convolucionais

As Redes Neurais Convolucionais (CNNs) são um dos tipos mais populares redes profundas. Esse tipo de rede possui ótimo desempenho para classificação de imagens, processamento de linguagem natural e visão computacional [4]. A principal vantagem de se utilizar uma CNN são suas características de ter uma estrutura simples, necessitar menos parâmetros de treinamento e sua capacidade de adaptação (generalização). Outro fator importante das CNNs são suas estruturas de compartilhamento de pesos, que além de tornarem esse tipo de rede mais similar com as rede neurais biológicas, faz com que a sua complexidade diminua, juntamente com a quantidade de pesos [5].

De acordo com 6, existem diversas arquiteturas de CNNs na literatura. Porém, os elementos básicos que compões esse tipo de rede neural são muito semelhantes. Basicamente uma CNN é composta de pelo menos três tipos de camadas: convolucional, *pooling* e camadas totalmente conectadas. Dentro das camadas convolucionais é onde ocorre o aprendizado das principais características da entrada fornecida, aplicando diversos filtros ou *kernel* e gerando mapas de características. Já quanto às camadas de *pooling*, estas são responsáveis por diminuir a resolução dos mapas de características e são comumente postadas entre camadas convolucionais. Por fim, após uma série de camadas convolucionais e de *pooling* estão as totalmente conectadas [1]

C. Classificação em conjuntos abertos

O problema de classificação em conjuntos abertos foi definido por 2, inicialmente pensando em problemas de visão computacional. Dentro deste contexto, um problema comum é a detecção de objetos, onde o objetivo é localizar um objeto conhecido dentro de uma imagem. Para isso, qualquer objeto que não seja o de interesse é considerado negativo, portanto, isto torna o problema muito mais aberto (com muitas classes de objetos negativos) do que fechado. No entanto, em abordagens comuns para detecção de objetos, classificadores binários são treinados com uma quantidade razoável de amostras positivas, juntamente com um número muito maior de amostras negativas. Porém, essas estratégias são apropriadas apenas quando existe uma boa amostragem de todas as classes negativas possíveis, caso contrário serão feitas classificações imprecisas.

Normalmente dentro de problemas tradicionais de classificação, como classificação binária, as classes são tratadas como positivas e negativas. Contudo, com os avanços nos estudos de classificação (especialmente dentro de conjuntos abertos) essa divisão entre as classes se tornou mais específica, como de acordo com 7, onde os autores afirmam que as classes se dividem em três categorias básicas:

1) *classes conhecidas*: classes de treinamento rotuladas como exemplos positivos (também servem como exemplos negativos para outras classes);
2) *classes conhecidas desconhecidas*: classes de treinamento rotuladas como exemplos negativos;
3) *classes desconhecidas desconhecidas*: classes não conhecidas na etapa de treinamento.

Posteriormente, foi adicionada uma categoria extra por 1, que diz respeito às *classes desconhecidas conhecidas*. Nesta última categoria existem classes desconhecidas durante a fase de treinamento, no entanto, estão disponíveis informações secundárias que podem ser utilizadas para o aprendizado. Essa categoria de classes é utilizada em uma técnica de aprendizado conhecida como *zero-shot learning*.

Em um cenário de classificação tradicional, apesar das amostras serem diferentes, as classes que serão levadas em consideração pelos classificadores são as mesmas classes vistas no treinamento e no teste. Por conta disso, todas as classes são conhecidas e isso configura um conjunto fechado. Em contrapartida, dentro do contexto de conjuntos abertos, além de classes conhecidas, durante a fase de teste surgirão *classes desconhecidas desconhecidas*. Deste modo, amostras pertencentes a esse tipo de classe deverão ser classificadas como desconhecidas, e não com a classe que apresente maior probabilidade, como ocorre em classificadores normais de conjunto fechado [7].

Dentro da área de aprendizado de máquina, o desempenho de uma função de previsão é medida pela frequência com que as previsões se diferem do valor real. Em outras palavras, isso pode ser entendido como a quantidade de vezes que um classificador realiza classificações incorretas. Com isso em mente, é necessário encontrar uma função que minimize a frequência de classificações erradas, conhecida como *risco empírico* [8]. De acordo com 1, existe um espaço longe dos dados conhecidos que é chamado de *espaço aberto*. Com isso em mente, classificar uma amostra proveniente desse espaço em uma classe conhecida incorre a um risco denominado *risco de espaço aberto*. Com esses conceitos definidos, o problema

[1] As camadas totalmente conectadas são opcionais caso seja implementada uma convolucional que produza uma saída de dimensão 1x1

de classificação em conjuntos abertos pode ser estabelecido como encontrar uma função de classificação que minimize o chamado *risco de conjunto aberto*, que por sua vez, é a combinação do risco empírico com o risco de espaço aberto [2].

O conjunto de dados é um fator impactante dentro do contexto da classificação em conjuntos aberto. Assim, 2 definiram o conceito de *grau abertura* de problema, que está relacionado ao quão aberto é um problema levando em consideração as classes presentes dentro do conjunto de dados. O *grau abertura* pode ser calculado pela Equação 1:

$$abertura = 1 - \sqrt{\frac{2 \times Tr}{Te + Ta}}, \tag{1}$$

onde Tr, Te, Ta representam, respectivamente, o número de classes de treinamento (*train*), teste (*test*) e número de classes alvo (*target*).

Posteriormente, 1 ajustaram a equação considerando que o número de classes alvo possui exatamente a mesma quantidade de classes do treinamento, resultando na Equação 2:

$$abertura = 1 - \sqrt{\frac{2 \times Tr}{Te + Tr}}, \tag{2}$$

onde Tr e Te representam o número de classes de treinamento e teste, respectivamente.

III. Trabalhos correlatos

A. Towards Open Set Deep Networks

Esta foi a primeira abordagem que utilizou aprendizado profundo no contexto de conjuntos abertos. Os autores propuseram um método onde utilizam *OpenMax* na camada de saída, sendo que esta, por sua vez, é uma extensão da função de ativação *Softmax*[2] e permite a classificação de uma amostra como desconhecida. Neste estudo, foi demonstrado que a função *OpenMax* reduz erros, como classificar imagens sem sentido em alguma classe conhecida, que comumente são cometidos por redes profundas. Além disso, utilizando essa técnica é possível limitar o risco de espaço aberto, portanto, ela fornece uma solução para o problema de conjuntos abertos [9]. Apesar de trazer uma solução válida para o problema, nesta abordagem existem alguns parâmetros que devem ser passados manualmente e que precisam ser escolhidos com cuidado de acordo com cada base de dados, o que nem sempre é viável.

B. Open Set Recognition Using Intra-Class Splitting

Neste trabalho, foi desenvolvido um método genérico para resolver o problema de classificação em conjuntos abertos utilizando aprendizado profundo. A ideia por trás dessa técnica é dividir as amostras de treino em dois subconjuntos: amostras típicas e atípicas. O primeiro subconjunto é relativo às amostras que possuem maior nível de similaridade com suas respectivas classes. Em contrapartida, o segundo conjunto é formado por amostras que possuem menos similaridade com suas classes [10]. Com base na premissa de que dentro de um espaço latente as amostras atípicas são mais parecidas com as classes desconhecidas em comparação com as classes típicas, este conjunto de amostras pode ser utilizado para fazer a modelagem das classes desconhecidas [11]. Para melhor entendimento, a Figura 1 ilustra uma divisão de amostras pertencentes a três classes conhecidas através das linhas pontilhadas. Note que as as classes possuem duas amostras, porém, uma de cada é atípica e por isso estas estarão no mesmo grupo e compartilharão a mesma classe com as amostras desconhecidas após o ICS (*intra-class splitting*).

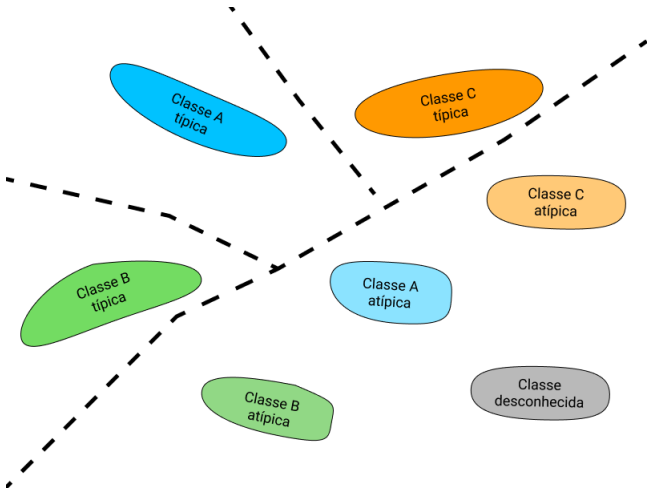

Fig. 1. Exemplo de divisão entre classes (ICS). As linhas azuis representam os limites entre cada classe. Note que, por exemplo, mesmo classe A possuindo duas amostras, uma delas é atípica, portanto ela compartilhará a mesma classe que as todas as outras amostras atípicas e também com as anormais (desconhecidas) [10].

Para fazer a divisão dos dados em amostras típicas e atípicas, uma rede neural é treinada com um número de K classes. Com a rede treinada, os dados são classificados e divididos com base em um *score* calculado conforme a seguinte definição: $f(.)$ é o mapeamento de uma rede neural de K classes e x representa uma amostra do conjunto de dados do treinamento. Deste modo, as probabilidades previstas no mapeamento são $\hat{\mathbf{y}}prob = f(x)$, onde $\hat{\mathbf{y}}prob \in \mathbb{R}^{K \times 1}$. $\hat{\mathbf{y}}$ é a classe resultante da previsão e é apresentado no valor de *one-hot encoding*. $\mathbf{y} \in \mathbb{R}^{K \times 1}$ indica o valor real da classificação dentro do *one-hot encoding* e $\odot$ representa o cálculo do produto de *Hadamard*. Por fim, o *score* para o ICS é denotado como demonstrado na Equação 3:

$$score = (\hat{\mathbf{y}}prob \odot \hat{\mathbf{y}} \odot \mathbf{y})^T \times \mathbf{1}, \tag{3}$$

onde $score \in \mathbb{R}$, $\mathbf{1} \in \mathbb{R}^{N \times 1}$, e $\mathbf{1}$ representa um vetor de números 1. Com uma taxa de *splitting* predefinida ρ, $\rho\%$ amostras com os *scores* mais baixos serão selecionadas para compor o grupo de amostras atípicas, enquanto as restantes serão consideradas típicas [10].

Com essa divisão de dados, subentende-se que uma rede profunda com $K + 1$ neurônios de saída é o suficiente para um classificador de conjuntos abertos. No entanto, as amostras

[2]Esta função produz o valor de saída de uma camada de neurônios totalmente conectada, produzindo uma distribuição de probabilidade em cima de N classes conhecidas

dos conjuntos atípicos normais possuem uma classificação diferente do que são na realidade, isto é, podem ser classificadas incorretamente no treinamento. Para evitar essa situação, uma sub-rede de regularização de conjunto fechado será utilizada para classificar estas amostras corretamente. Ao final, esse classificador terá duas saídas, sendo que uma delas é responsável pela classificação de conjunto aberto e a outra para conjunto fechado [10].

Apesar desse método apresentar bons resultados em comparação a outros trabalhos do estado da arte, seus experimentos foram realizados em bases de dados utilizadas para fins didáticos. No presente trabalho, será avaliada a efetividade do método de classificação de conjuntos abertos com ICS utilizando bases de dados que estejam relacionadas com problemas do mundo real, com o intuito de buscar uma solução que possa ser utilizada em outras aplicações que vão além de estudos da área acadêmica.

IV. Metodologia

A. Visão geral

Para abordar o problema deste trabalho, propõe-se a utilização do método de *Intra-Class Splitting* para modelagem das classes desconhecidas e conhecidas, proposta por 10, dentro do contexto de classificação de conjuntos abertos. De forma geral, o método apresentado é dividido em três passos principais, onde a saída de cada um é a entrada para o próximo passo: *Preparação dos dados*, *Treinamento dos modelos* e *Avaliação dos modelos*. A Figura 2 ilustra o fluxo da metodologia, onde cada passo é representado por um bloco pontilhado. Com isso definido, cada passo possui suas próprias particularidades que serão abordados detalhadamente mais adiante.

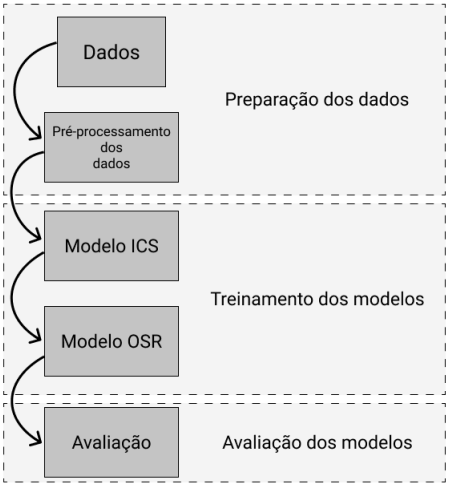

Fig. 2. Visão geral do método

B. Preparação dos dados

Na etapa de preparação dos dados é onde ocorrem todas as tratativas em relação aos dados que serão submetidos aos modelos. Especificamente, é neste passo onde os dados da base selecionada são convertidos para o mesmo padrão, com o intuito de deixá-los no formato de entrada da arquitetura dos modelos. Caso necessário, é também nesta etapa onde podem ser realizadas as normalizações dos dados.

A respeito das classes das amostras, elas devem estar no formato de *one-hot enconding* (exemplo: [0, 0, 1, 0]). Isto se dá pela necessidade deste formato de matriz que é utilizado pelos cálculos envolvidos no método proposto.

Como observado na Figura 2, a saída produzida pela *Preparação dos dados* será a entrada para o próximo passo, que é o *Treinamento dos modelos*. Portanto, durante a etapa atual, é de suma importância que se faça a divisão dos dados para treinamento e teste, com objetivo de obter resultados com maior confiabilidade no passo seguinte.

C. Treinamento dos modelos

Dentro do método proposto são necessários dois modelos de redes neurais artificiais profundas. O primeiro modelo, chamado neste trabalho de *modelo ICS*, é responsável pela divisão das amostras da base de dados do problema em típicas e atípicas utilizando (*Intra-class Splitting*). Já o segundo modelo, denominado *modelo OSR* [3], é responsável pela classificação de conjunto aberto em si, sendo que ele é treinado com os dados divididos pelo *modelo ICS*.

D. Avaliações

O intuito desta é etapa é fazer a avaliação do *modelo ICS*, e também do *modelo OSR*. Para isso, as previsões feitas na fase de teste de cada modelo serão submetidas à métricas para verificar a performance e, por consequência, avaliar a qualidade dos modelos quando postos à prova em dados não vistos no treinamento.

V. Experimentos e resultados

A. Tecnologias

As duas redes neurais previstas no desenvolvimento neste trabalho serão feitas utilizando *Keras*, uma API de aprendizado profundo escrita em *Python* e que roda em cima da plataforma de aprendizado de máquina conhecida como *TensorFlow*. O *Keras* fornece diversas abstrações feitas em cima do *TensorFlow* que facilitam o desenvolvimento de RNAs [12].

No que diz respeito aos modelos, serão criadas duas redes neurais convolucionais, tendo em vista que as entradas serão imagens: uma para o *modelo ICS* e outra para o *modelo OSR*. Será utilizada a mesma arquitetura nas duas redes, com uma única diferença no que diz respeito à camada do *modelo ICS*: esta terá um neurônios de saída a menos, tendo em vista que algumas classe serão consideradas desconhecidas e compartilharão o mesmo valor.

Os cálculos para avaliação dos resultados obtidos serão feitos com o auxílio de um módulo de *Python* chamado *Scikit-learn* [13]. Este módulo possui, além de ferramentas para aprendizado de máquina, diversas funções que podem ser utilizadas para a análise de dados.

[3]OSR vem do inglês *open-set recognition*, termo da literatura internacional para classificação/reconhecimento de conjuntos abertos.

B. Base de dados

Os problemas de visão computacional regularmente se encaixam dentro do contexto de conjuntos abertos. Por conta disso, a base de dados utilizada nos experimentos é uma base de imagens. Este conjunto de dados está relacionado com previsão do tempo, e é composto de mais de 1500 imagens RGB, contendo diversas imagens com o tempo chuvoso, nublado, ensolarado, enevoado e também do nascer do sol.

A base de dados original [4] já está dividida em treinamento/teste. No entanto, para aproveitar todas as amostras na etapa de ICS, estes dados foram unificados para que posteriormente sejam divididos de acordo com a necessidade. Como existem diversas imagens com resoluções diferentes, é importante que todas elas sejam transformadas para o mesma resolução. Por conta disso, cada imagem da base de dados foi redimensionada para 224 pixels de altura, 224 de largura, com 3 canais de cores (RGB), portanto, os modelos terão como entrada uma estrutura de dados com dimensão 224x224x3.

VI. Tecnologias

Todos os pré-processamentos e experimentos previstos serão desenvolvidos na plataforma *Google Colab*, pois além de gratuita, ela oferece recursos computacionais, como GPUs, que aceleram o treinamento das redes neurais artificiais.

As duas redes neurais artificias serão construídas utilizando o *Keras*, uma API de aprendizado profundo desenvolvida em *Python* que roda em cima da plataforma de aprendizado de máquina conhecida como *TensorFlow*. O *Keras* fornece diversas abstrações feitas em cima do *TensorFlow* que facilitam o desenvolvimento de RNAs, além de funções para pré-processamento de dados, entre outros recursos [12].

No que diz respeito aos modelos, serão criadas duas redes neurais convolucionais, tendo em vista sua boa performance com imagens. Uma rede será para o *modelo ICS* e outra para o *modelo OSR*. Será utilizada a mesma arquitetura nas duas redes, com uma única diferença em relação à camada de saída do *modelo ICS*: esta terá neurônios de saída a menos, tendo em vista que algumas classe serão consideradas desconhecidas e compartilharão o mesmo valor.

Os cálculos para avaliação dos resultados obtidos serão feitos com o auxílio do módulo escrito em *Python* chamado *Scikit-learn*. Este módulo possui, além de ferramentas para aprendizado de máquina, diversas ferramentas que podem ser utilizadas para a análise de dados [13].

A. Métricas de avaliação

A acurácia balanceada (BACC) [14] será uma das medidas para avaliar o desempenho do modelo, pois ela fornece uma comparação justa entre bases de dados balanceadas e desbalanceadas, o que é muito comum no contexto de conjuntos abertos. Para isso, a BACC pode ser calculada de acordo com a Equação 4:

$$BACC = \frac{1}{2} \times (\frac{TP}{TP+FN} + \frac{TN}{FP+TN}), \qquad (4)$$

[4]https://www.kaggle.com/vijaygiitk/multiclass-weather-dataset

onde TP representa os verdadeiros positivos, FN os falsos negativos, TN os verdadeiros negativos e FP os falso positivos.

Para complementar a avaliação dos resultados, a matriz de confusão será analisada juntamente da área abaixo da curva ROC para ambos os modelos. A acurácia também será uma das medidas utilizadas para analisar os resultados obtidos pelas duas redes neurais.

B. Experimento 1: desenvolvimento do modelo ICS

O intuito deste experimento é desenvolver um modelo capaz de classificar todas as amostras da base de dados em suas respectivas classes, para posteriormente, quando o mesmo obter um bom desempenho de classificação, seja utilizado para dividir as amostras de acordo com o método de *Intra-class splitting* de **?**. Com base nisso, é esperado que ao final do processo o modelo apresente pelo menos 85% de BACC e 90% no valor da área abaixo da curva ROC.

O experimento foi conduzido variando o parâmetro de épocas de treinamento utilizando a arquitetura da Figura 3, com 5 neurônios de saída. Para aumentar a quantidade de dados de treinamento, foi utilizada a classe *ImageDataGenerator* do *Keras*. Em relação aos dados, foram selecionados 80% para treinamento e 20% para teste. Por fim, o modelo que alcançou a melhor performance foi treinado em 300 épocas com *batch size* de 128 e seus resultados podem ser observados na Tabela I.

```
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(output, activation='softmax'))
```

Fig. 3. Arquitetura do *modelo ICS*

TABLE I
Resultados de teste do *modelo ICS*

Modelo	BACC	ACC	AUC ROC
ICS	0.9096	0.9117	0.989

Como observado nos resultados, o modelo superou em aproximadamente 5% a BACC esperada e em quase 9% a área abaixo da curva ROC. Portanto, os resultados comprovam a eficiência do modelo para que se possa prosseguir para a próxima etapa, que se trata de realizar o ICS.

C. Experimento 2: Intra-class splitting

Utilizando o *modelo ICS* desenvolvido anteriormente, este experimento será responsável por dividir a base de dados com ICS. Com base no valor calculado pela Equação 3, a

expectativa é que, ao final, aproximadamente 5% das amostras com menor *score* sejam agrupadas no conjunto de amostras atípicas e as remanescentes no conjunto de amostras típicas.

Para conduzir este experimento, 1528 amostras foram submetidas ao processo de ICS com o parâmetro $\rho = 10$. Após o cálculo do *score*, 153 amostras (aproximadamente 10.01%) obtiveram os menores valores, portanto, foram definidas como atípicas. As outras 1375 ficaram para o grupo das típicas. Com essa divisão, tem-se o necessário para o desenvolvimento do *modelo OSR*.

D. Experimento 2: Intra-class splitting

Utilizando o *modelo ICS* desenvolvido anteriormente, este experimento será responsável por dividir a base de dados com ICS. Com base no valor calculado pela Equação 3, a expectativa é que, ao final, aproximadamente 5% das amostras com menor *score* sejam agrupadas no conjunto de amostras atípicas e as remanescentes no conjunto de amostras típicas.

Para conduzir este experimento, 1528 amostras foram submetidas ao processo de ICS com o parâmetro $\rho = 10$. Após o cálculo do *score*, 153 amostras (aproximadamente 10.01%) obtiveram os menores valores, portanto, foram definidas como atípicas. As outras 1375 ficaram para o grupo das típicas. Com essa divisão, tem-se o necessário para o desenvolvimento do *modelo OSR*.

TABLE II
RESULTADOS DE TESTE DO *modelo OSR*

Modelo	BACC	ACC	AUC ROC
ICS	0.7602	0.658	0.8533

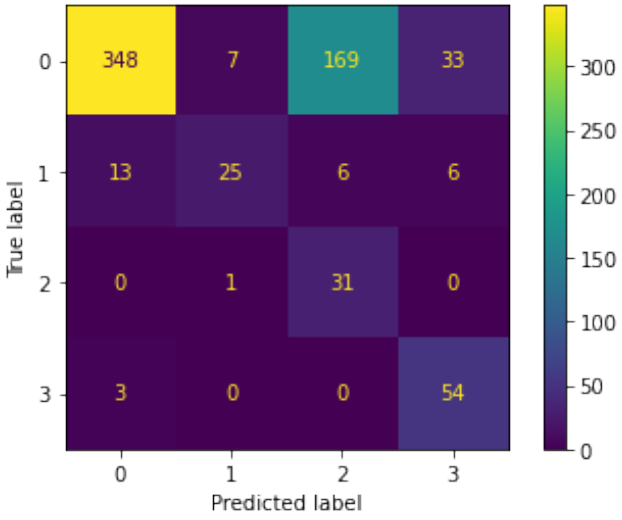

Fig. 4. Matriz de confusão do *modelo OSR*

Como esperado, o *modelo OSR* conseguiu atingir aproximadamente 76% de BACC, superando em 6% a expectativa inicial. No entanto, é importante destacar a quantidade de erros da classe desconhecida, que de um total 557 previsões, 209 foram desacertos, dando ênfase para a classe 2, que foi prevista 169 vezes erroneamente como desconhecida (classe 0). Deste modo, é possível avaliar que os resultados podem ser melhorados, levando em consideração que bases de dados maiores podem ajustar melhor o modelo, fazendo com que ele seja mais preciso, assim como alterar a arquitetura utilizada.

VII. CONCLUSÃO

O presente trabalho propôs a utilização de aprendizado profundo, combinado com a abordagem de *intra-class splitting*, como ferramenta para resolver o problema de classificação de conjuntos abertos. Este problema em questão é muito comum em problemas do mundo real, onde tem-se uma grande quantidade de dados e, muitas vezes, não é possível rotular todos. A partir disso, surge-se a necessidade dos modelos de aprendizado de máquina trabalharem lidando com dados desconhecidos durante o treinamento.

Dentro do método proposto, o desconhecido é modelado através de amostras atípicas, que são aquelas que possuem menor semelhança à suas respectivas classes. Para isso, é realizado o cálculo formalizado no trabalho dos autores do método de *intra-class splitting*. Sendo assim, o método deste trabalho foi testado em uma base de dados de previsão do tempo, que foi escolhida com o intuito de evitar bases de dados do meio acadêmico, que geralmente são escolhidas para avaliar novas técnicas, como acontece no trabalho em que o presente método foi baseado.

Assim sendo, neste trabalho foi possível diferenciar a classificação tradicional da classificação de conjunto aberto, bem como sugerir uma metodologia para o contornar o problema abordado, realizar experimentos e avaliar os resultados alcançados. Em vista disso, os resultados obtidos demonstram que o metodologia proposta é promissora e capaz de resolver o problema de classificação de conjuntos abertos. Além do mais, podem ser feitos ajustes, como aumentar ainda mais os dados, refinar redes neurais pré-treinadas, sempre visando obter melhor performance de classificação.

Com relação a trabalhos futuros, pode-se utilizar o método sugerido em outras aplicações além da classificação de imagens. Para isso, basta analisar o tipo de dados do problema e desenvolver os modelos com base na necessidade em pauta. Mais além, também pode-se estender o trabalho para reconhecimento de mundo aberto, como foi abordado por 1. Ademais, comparações com outros métodos utilizando outras bases de dados são igualmente válidas, a fim de possibilitar o estudo de alternativas para resolver o problema de classificação de conjuntos abertos.

REFERENCES

[1] C. Geng, S.-J. Huang, and S. Chen, "Recent advances in open set recognition: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[2] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult, "Towards open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 35, July 2013.

[3] X. Wang, Y. Zhao, and F. Pourpanah, "Recent advances in deep learning," *International Journal of Machine Learning and Cybernetics*, vol. 11, Fevereiro 2020.

[4] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*. Manhattan: IEEE, 2017, pp. 1–6.
[5] T. Liu, S. Fang, Y. Zhao, P. Wang, and J. Zhang, "Implementation of training convolutional neural networks," 2015.
[6] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320317304120
[7] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 36, nov 2014.
[8] X.-D. Zhang, *Machine Learning*. Singapore: Springer Singapore, 2020, pp. 223–440.
[9] A. Bendale and T. E. Boult, "Towards open set deep networks," *CoRR*, vol. abs/1511.06233, 2015. [Online]. Available: http://arxiv.org/abs/1511.06233
[10] Y. L. Patrick Schlachter and B. Yang, "Open-set recognition using intra-class splitting," in *2019 IEEE European Signal Processing Conference (EUSIPCO)*, September 2019.
[11] P. Schlachter, Y. Liao, and B. Yang, "One-class feature learning using intra-class splitting," *CoRR*, vol. abs/1812.08468, 2018.
[12] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *CoRR*, vol. abs/1201.0490, 2012.
[14] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *2010 20th International Conference on Pattern Recognition*. Manhattan: IEEE, 2010, pp. 3121–3124.