Universidade Federal de Mato Grosso – UFMT
Campus Universitário de Várzea Grande
Faculdade de Engenharia
Engenharia de Computação

Lucas Steffens de Oliveira

DETECÇÃO DE CÂNCER DE PELE UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Várzea Grande
2021

Lucas Steffens de Oliveira

DETECÇÃO DE CÂNCER DE PELE UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.
Orientador: Prof. Dr. Raoni Florentino da Silva Teixeira

Várzea Grande
2021

Agradecimentos

Agradeço primeiramente aos meus pais Lisete Steffens e Lourival de Oliveira, meus irmãos Leandro e Luis, que sempre estão ao meu lado me apoiando e inspirando.

Ao meu orientador pelos ensinamentos durante a graduação e por aceitar conduzir este trabalho.

Ao professor Diogo pelos conselhos, professora Gracyeli e professor César pela oportunidade de participar dos projetos de pesquisa.

Aos meu colegas de curso que ao decorrer do tempo se tornaram amigos.

A todo brasileiro que financiou de forma indireta meus estudos.

Resumo

Steffens Oliveira, Lucas. **DETECÇÃO DE CÂNCER DE PELE UTILIZANDO REDES NEURAIS CONVOLUCIONAIS**. 96 p. Trabalho de Conclusão de Curso – Engenharia de Computação, Faculdade de Engenharia, Universidade Federal de Mato Grosso, Brasil, 2021.

O câncer de pele é o câncer mais recorrente no Brasil e sua taxa de letalidade tem aumentado ao decorrer dos anos no país. Por outro lado, caso detectado precocemente tem uma alta taxa de sobrevivência, portando, criar meios que auxiliem e acelerem o processo de diagnostico é primordial. Este trabalho utilizou de transferência de aprendizado para realizar a classificação de imagens de lesões de pele um possível câncer de pele, utilizando a base de dados ISIC reamostrada pelo método SMOTE. Os resultados obtidos indicaram que modelos criados através da inteligência artificial podem ser promissores para aceleram a detecção do câncer de pele.

Palavras-chave: Lesões de pele. Inteligência artificial. Transferência de Aprendizado. SMOTE. Máquina de vetores de suporte. .

Abstract

Steffens Oliveira, Lucas. **DETECÇÃO DE CÂNCER DE PELE UTILIZANDO REDES NEURAIS CONVOLUCIONAIS**. 96 p. Undergraduate Dissertation – Computing Engineering, Engineering Faculty, Federal University of Mato Grosso, Brazil, 2021.

Skin cancer is the most recurrent cancer in Brazil and its lethality rate has increased over the years in the country. On the other hand, if detected early, it has a high rate of technologies, therefore, creating means to assist and accelerate the diagnostic process is paramount. This work uses the download of learning to carry out a classification of skin images of a possible skin cancer, using an ISIC database resampled by the SMOTE method. The results obtained indicated that models created through artificial intelligence can be promising to accelerate the detection of skin cancer.

Keywords: Skin lesions. Artificial intelligence. Transfer Learning. SMOTE. Support vector machine..

Lista de ilustrações

Lista de tabelas

Sumário

Capítulo 1

Introdução

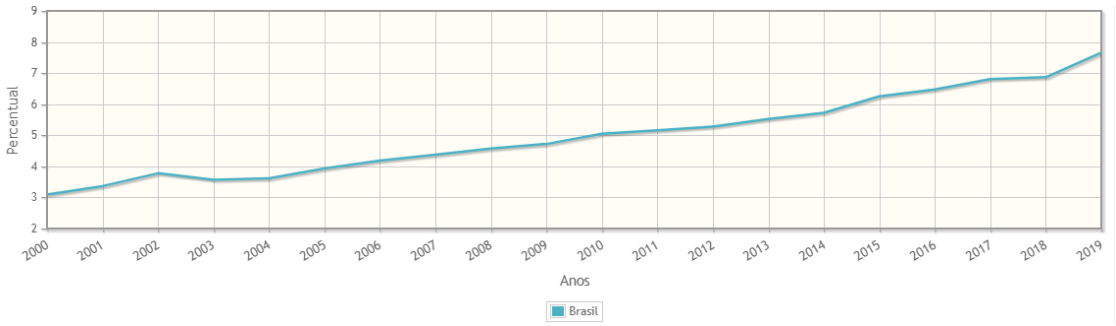
O câncer de pele é a enfermidade cancerígena mais recorrente no Brasil e no mundo. Segundo dados do Ministério da Saúde do Brasil, a doença é normalmente diagnosticada em indivíduos com idade acima de 40 anos e é incomum em crianças e pessoas negras (BRASIL, 2019).

O câncer é resultado da proliferação descontrolada de células que invadem os tecidos e órgãos, formando tumores. No caso específico do câncer de pele, os tumores são classificados em melanomas e não melanomas. O tumor do tipo melanoma, se origina nas células que são responsáveis pela produção de melanina (INCA, 2021), proteína responsável pela coloração da pele, portanto, ele pode ocorrer em qualquer parte da corpo, normalmente, surgem em forma de pintas escuras com bordas irregulares, sinais e manchas, que podem ou não apresentar coceira (INCA, 2021).

O Instituto Nacional do Câncer (INCA) estima que anualmente 8.450 novos casos surgem no Brasil, sendo 4.200 homens e 4.250 mulheres (INCA, 2021). A letalidade anual deste tipo de câncer é cerca de 1.978, sendo 1.159 homens e 819 mulheres (ATLAS, 2019). O câncer de pele do tipo não melanoma é bem mais comum do que o do tipo melanoma e representa 30% de todos tumores malignos diagnosticados no Brasil (INCA, 2021). Anualmente, são registrados em média 176.930 casos de câncer não melanoma, sendo 83.770 homens e 93.160 mulheres (INCA, 2021). A letalidade é menor e estima-se que anualmente 2.616 mortes ocorrem no Brasil, sendo 1.488 homens e 1.128 mulheres (ATLAS, 2019).

Mesmo a letalidade sendo considerada baixa, nas últimas décadas, houve um aumento de registro de mortes por neoplasias malignas ligadas a pele, como pode ser observado no gráfico da Figura 1.

Figura 1 – Gráfico proporcional do total de mortes por melanoma maligno de pele e outras neoplasias malignas de pele.



Fonte: (ATLAS, 2020)

A Figura 1 apresenta o percentual de mortes entre 2000 a 2019 no Brasil, considerando ambos os sexos e todas as idades. Nesse gráfico, é possível perceber um aumento de $4,57\%$ entre os anos de 2000 e 2019.

Apesar deste aumento, os cânceres de pele têm alta taxa de cura e sobrevivência relativa de cinco anos a partir do primeiro diagnóstico, dependendo do estágio que é encontrado. Sendo que para os melanomas em fase inicial localizados, a taxa de cura é de 99% (SOCIETY, 2019), e para os não melanomas em fase inicial localizados a taxa é de 100% (SOCIETY, 2017).

Dessa forma, se a enfermidade for diagnosticada de forma precoce, a chance de sobrevivência é alta. Sob essa perspectiva, a disponibilização e popularização de meios que indiquem a possível existência do câncer de pele, irão acelerar o processo de diagnóstico e, possivelmente, aumentar a chance de cura.

Um exemplo de sucesso de plataforma de saúde é a parceria do Google com o hospital Albert Enstein. Em 2016, as pesquisas com termos diretamente relacionados a saúde e buscas de informações sobre doenças e sintomas, correspondiam a 5% de todas as buscas do Google (EINSTEIN, 2016), e para garantir a credibilidade do retorno destas buscas, o buscador disponibiliza um quadro com as possíveis enfermidades, descrições, variações, sintomas e a necessidade de diagnóstico médico. O corpo clínico do hospital valida as informações a serem exibidas.

Detectar atipias, através da análise de imagens dermatoscópicas, não é uma tarefa fácil até mesmo para os dermatologistas. Em uma pesquisa realizada em (TSCHANDL et al., 2017), 26 dermatologistas foram expostos a 59 quartetos de imagens, onde cada um continha um melanoma e três outras lesões de pele. Após a avaliação dos especialistas selecionados, a média de detecção foi de apenas 24 dos 59 melanomas (ou seja, aproximadamente, 40% das imagens dos melanomas foram classificados como lesões ou nevos atípicos). Outra dinâmica semelhante realizada em (ESTEVA et al., 2017), comparou a precisão de classificação de um modelo de rede neural com a de dois dermatologistas, em

dois cenários diferentes: a) com imagens de lesões benignas, malignas e não neoplásicas e b) onde subclassificações das três classes do experimento anterior eram avaliadas. No primeiro cenário, os dermatologistas obtiveram taxas de acerto de 65,56% e 66,0% e, no segundo, as taxas foram de 53,3% e 55,0%, respectivamente.

Inspirados por esses resultados, propomos nesse trabalho um método de detecção do câncer para médicos especializados. Nossa hipótese de pesquisa é que modelos preditivos que utilizem imagens para classificar uma lesão na pele, como um possível candidato a câncer, podem acelerar o processo de diagnóstico, através da popularização do acesso à informação médica. Esse tipo de método pode oferece um indicativo a mais para profissionais solicitarem biópsias.

Este trabalho tem como base a pesquisa de (ESTEVA et al., 2017) e vale ressaltar que todos materiais resultantes são apenas para fins informativos e não substituem o diagnóstico médico, aconselhamento ou tratamento especializado.

1.1 Objetivo Geral

Este trabalho tem o objetivo de realizar a classificação de imagens de lesões dermatoscópicas, em possíveis cânceres de pele, utilizando redes neurais artificiais.

1.2 Objetivos Específicos

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

a) Obter uma base de dados de lesões benignas e malignas, sem duplicatas.

b) Obter bases de treinamento reamostradas através técnica de sobreamostragem minoritária (SMOTE).

c) Obter modelos de classificação de lesões de pele.

d) Comparar os resultados dos modelos com os encontrados na literatura.

1.3 Organização do Texto

Este trabalho está organizado em 7 capítulos, sendo o capítulo 2 responsável por uma breve revisão bibliográfica; o capítulo 3 apresenta os materiais e métodos utilizados; o capítulo 4 apresenta as etapas de processamento e análise das imagens; o capítulo 5 trata sobre a classificação das imagens; o capítulo 6 apresenta os resultados do experimentos realizados; o capítulo 7 fala sobre as conclusões e possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta uma breve revisão bibliográfica sobre as técnicas de Inteligência Artificial aplicadas neste estudo.

2.1 Inteligência Artificial

A Inteligência artificial (IA), é uma área da ciência e engenharia, que originou-se após a segunda guerra mundial, e teve seu nome definido em 1956 (RUSSELL; NORVIG, 2013). Segundo (HAENLEIN; KAPLAN, 2019) a inteligência artificial se refere a sistemas capazes de aprender com dados para atingir objetivos específicos através da adaptação, que sejam capazes de definir dados externos de forma correta.

De maneira um pouco mais usual, o objetivo da inteligência artificial é definir algoritmos que tenham um desempenho semelhante ao de um humano ao realizar uma determinada tarefa. Desde sua origem, a inteligência artificial já se subdividiu em diversas áreas e subáreas especializadas, uma delas é a visão computacional.

A visão computacional é uma área da inteligência artificial que procura repassar para máquinas a capacidade da visão (BACKES; JUNIOR, 2016). O termo visão se refere a capacidade de analisar imagens, retirar informações, interpretá-las e classificá-las.

Desta forma, os sistemas de visão são implementados com a capacidade de adquirir imagens, processar, segmentar, extrair características e reconhecer padrões, podendo conter, menos ou mais fases a depender do objetivo do sistema (BACKES; JUNIOR, 2016).

As redes neurais convolucionais são redes empregadas para realizar tarefas de visão computacional, um exemplo de aplicação de redes convolucionais é o Google Lens, que oferece o serviço de pesquisa, identificação e classificação de objetos, plantas e animais através de imagens (LENS, 2021). Da mesma empresa, outro exemplo é o serviço de pequisa por pessoas, coisas e lugares da galeria de imagens em nuvem Google Fotos (FOTOS, 2021).

Ademais, esta arquitetura de rede é utilizada para diversas outras aplicações como reconhecimento facial (PARKHI; VEDALDI; ZISSERMAN, 2015), classificação de ima-

gens médicas (YADAV; JADHAV, 2019), tradução de textos (LENS, 2021), detecção de objetos para carros autônomos (PENG et al., 2020) entre outras. Neste estudo, esta arquitetura de rede foi utilizada.

2.2 Redes Neurais Convolucionais

As redes neurais convolucionais (CNN) são o tipo de rede mais utilizada na área da visão computacional, por sua arquitetura possuir a característica de processar imagens e vídeos e ser eficaz em tarefas de classificação. Segundo (BROWNNLEE, 2017):

> As redes neurais convolucionais esperam e preservam a relação espacial entre os pixels, aprendendo representações de recursos internos usando pequenos quadrados de dados de entrada. O recurso é aprendido e usado em toda a imagem, permitindo que os objetos nas imagens sejam deslocados ou traduzidos na cena e ainda detectáveis pela rede.

Uma CNN é qualquer rede neural que utiliza de convolução em uma de suas camadas. Uma rede deste tipo, normalmente, é formada por três tipos de camadas, as convolucionais, de pool e as totalmente conectadas.

2.2.1 Camadas Convolucionais

Estas camadas são as responsáveis por realizar a operação matemática de convolução com as imagens, segundo (OGATA, 2010) a convolução é matematicamente definida como:

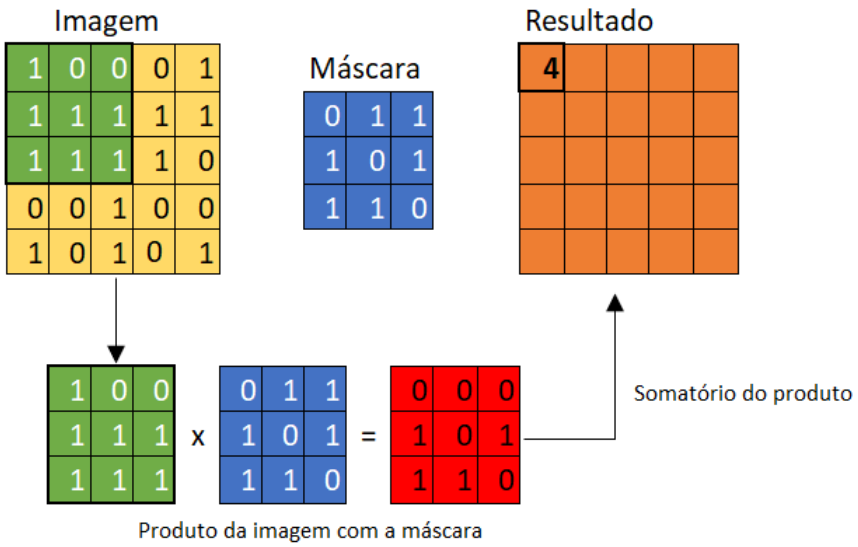
$$M(x) * I(x) = \int_0^t M(t)I(x-t)dt \tag{1}$$

Porém, ao se trabalhar com imagens, a operação é realizada em sua forma discreta. Seja uma imagem I e uma máscara (filtro) M, (BACKES; JUNIOR, 2016) define a convolução discreta para duas dimensões como:

$$M(x,y) * I(x,y) = \sum_{s=-a}^{a} \sum_{t=-l}^{l} M(s,t)I(x-s, y-t) \tag{2}$$

Onde a largura e a altura da máscara M são representadas por a e l, respectivamente. Segundo (BACKES; JUNIOR, 2016), podemos definir esta operação como uma técnica que combina a intensidade de um conjunto de pixeis vizinhos em um novo pixel, gerado através da imagem original. Este filtro é aplicado percorrendo toda a imagem, no fim da operação, geramos uma nova imagem derivada da original, que normalmente são denominadas como matriz ou mapa de característica. A Figura 2 apresenta o diagrama de uma iteração do processo de aplicação de uma máscara em uma imagem por convolução.

Figura 2 – Processo de uma aplicação de máscara em uma imagem por convolução.



Fonte: O autor.

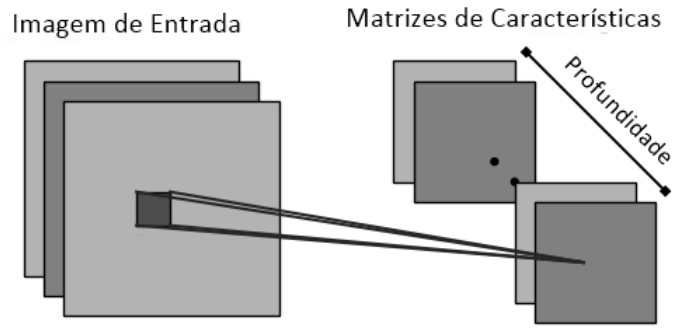
Este processo é realizado através de toda matriz da imagem, em todos canais que a constituem. Após a operação, a matriz resultante passa por uma função de ativação, que aplica uma função matemática no mapa de característica para controlar quais neurônios da próxima camada serão ativados (NWANKPA et al., 2020), segundo (AMIDI; AMIDI, 2018) a mais comum nos volumes de saídas das convoluções é a ReLU.

Normalmente nas camadas de convolução, são aplicados vários filtros, estes que geram varias matrizes de características a partir da imagem original, que durante o processamento pode tornar-se um problema, pois aumentará o consumo de recursos como memória e tempo de processamento.

Para controlar o tamanho da saída das camadas de convolução, sem abdicar de aplicar vários filtros, segundo (STANFORD, 2015), devemos utilizar os seguintes hiperparâmetros: profundidade (depth), padding e passo (stride).

A profundidade é diretamente proporcional a quantidade de filtros utilizados, portanto, quanto mais filtros mais profunda a saída de uma camada de convolução e mais mapas de características derivadas de uma imagem serão obtidas.

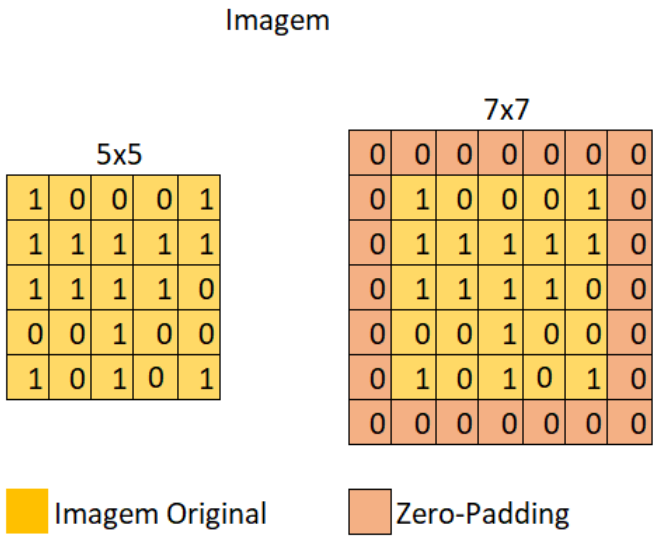
Figura 3 – Representação da profundidade.



Fonte: O autor.

O padding é a ação de preencher a borda da imagem de entrada com algum valor, normalmente, o zero (zero-padding), podendo assim, controlar a largura e altura da saída da convolução, uma vez que, o preenchimento da borda afeta diretamente como o filtro irá percorrer a imagem e a matriz característica resultante.

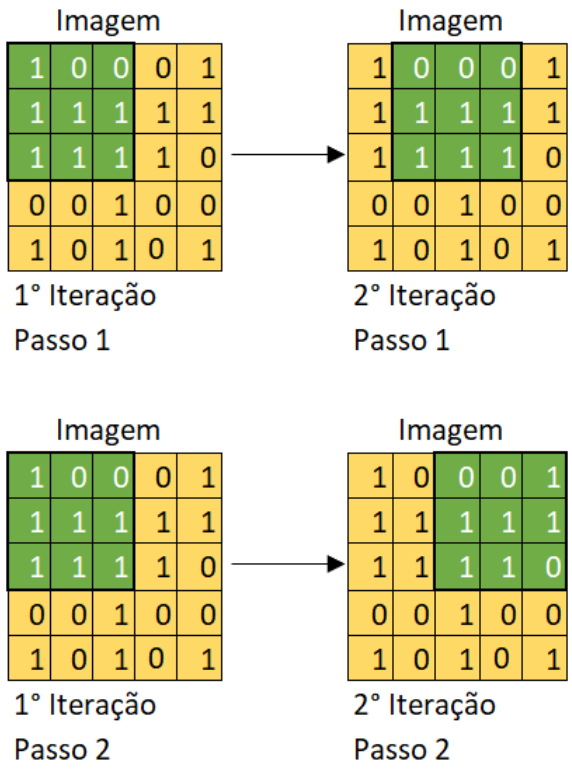
Figura 4 – Exemplificação da utilização do Zero-Padding.



Fonte: O autor.

O passo (stride) é o parâmetro que indica o quanto o filtro se movimentará na imagem após uma operação, ao definir o passo diferente de um, o filtro percorre a imagem se movimentando mais de um pixel por vez, como consequência temos uma imagem de saída com largura e altura menor que a original.

Figura 5 – Exemplificação do Stride durante as interações de convolução.



Fonte: O autor.

Conforme (STANFORD, 2015) com estes hiperparâmetros, podemos controlar os tamanhos dos volumes de saída das camadas convolucionais com a seguinte fórmula:

$$DMC = ((D - F + 2P)/S) + 1 \tag{3}$$

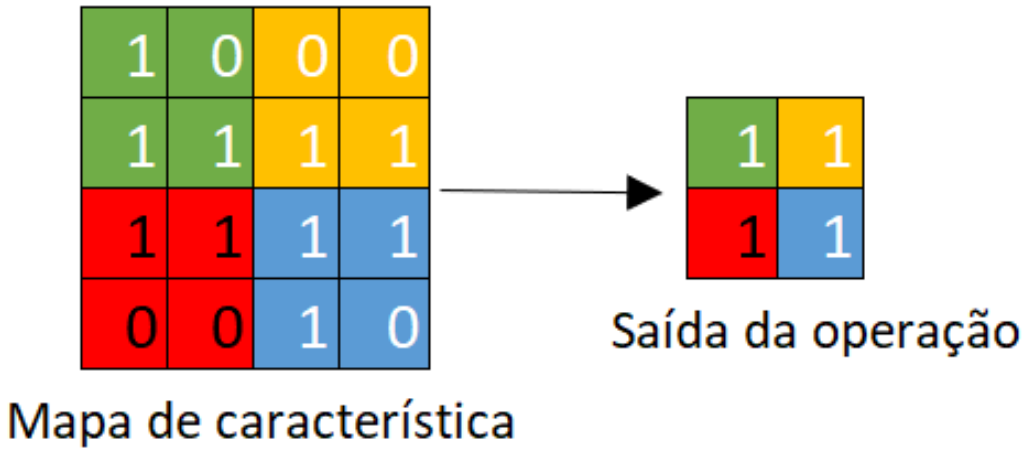
Onde DMC é a dimensão do mapa de característica, D a dimensão da imagem de entrada, F o tamanho do filtro, S o stride e P o padding.

2.2.2 Camada de Pool

As camadas de pool, tal qual os hiperparâmetros supracitados, tem a função de realizar a redução da dimensão dos volumes de saída das camadas convolucionais ao aplicar uma função que preserve as principais características da imagem. Por este motivo, normalmente, temos uma camada de pool após uma camada convolucional. O pooling se assemelha a aplicação dos filtros, porém, diferentemente da operação de convolução, é aplicada uma função nos elementos da matriz.

Seja uma matriz de característica 4x4 e um max-pooling (função de maximização) 2x2 com stride dois, temos:

Figura 6 – Exemplificação do Max-Pooling.



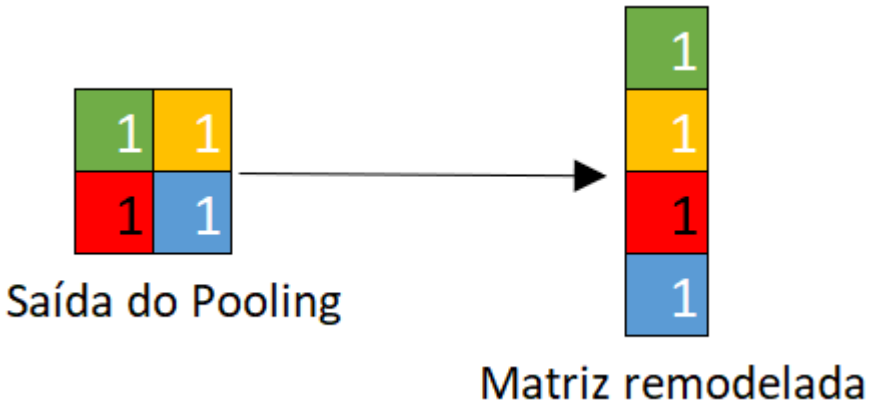
Fonte: O autor.

Onde cada cor na matriz do mapa de característica representa uma operação da função de max-pooling e o resultado é o equivalente a cor na matriz de saída da operação. Esta operação é realizada em todos mapas de características da saída da camada de convolução, e desta forma, os valores menos significativos são eliminados e a dimensão diminuída, assim, acelerando o processamento.

2.2.3 Camadas Totalmente Conectadas

As camadas totalmente conectadas são as responsáveis por realizar de fato a classificação das imagens de entrada através das informações disponibilizadas pelas camadas anteriores, para isto, as matrizes de características são transformadas em vetores.

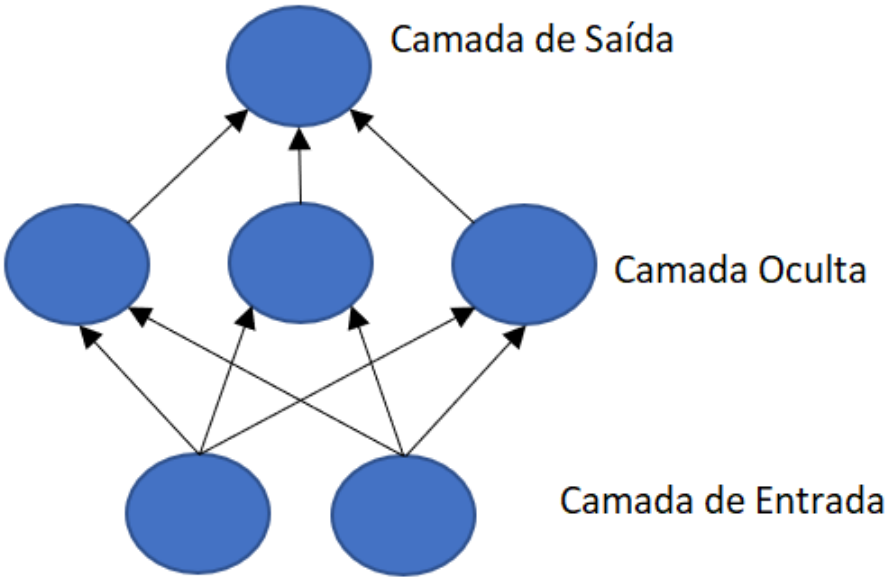
Figura 7 – Flatten da matriz de característica para a entrada nas camadas totalmente conectadas.



Fonte: O autor.

Após a remodelagem, este vetor é inserido na camada totalmente conectada, que consiste em uma rede de várias camadas constituídas por neurônios.

Figura 8 – Rede Multicamada.



Fonte: O autor.

Onde cada círculo azul representa um neurônio e as setas suas conexões. Os neurônios são as unidades básicas das redes neurais, que segundo (BROWNNLEE, 2017), são unidades computacionais que recebem sinais de entrada e geram sinais de saída usando uma função de ativação.

Figura 9 – Modelo de um neurônio.

Fonte: O autor.

Estes que podem ser matematicamente definidos, segundo (RUSSELL; NORVIG, 2013), como:

$$g(in_j) = g(\sum_{i=0}^{n} w_{i,j} a_i) \tag{4}$$

Cada linha de neurônios é definido como uma camada, e a junção de todas as camadas formam uma rede perceptron multicamada. Na última camada MPL teremos uma camada formada com o mesmo número de neurônios da quantidade de classes que a rede pode classificar.

De forma resumida, uma camada totalmente conectada recebe entradas (vetores) que são multiplicadas por pesos e o resultado passa por uma função de ativação. A cada época os pesos dessa rede são ajustados utilizando o erro encontrado na saída da rede. No final após uma função de ativação essas camadas retornam a probabilidade de classificação da entrada avaliada.

Outra maneira clássica de realizar classificação utilizando vetores como entrada é através de uma máquina de vetores de suporte.

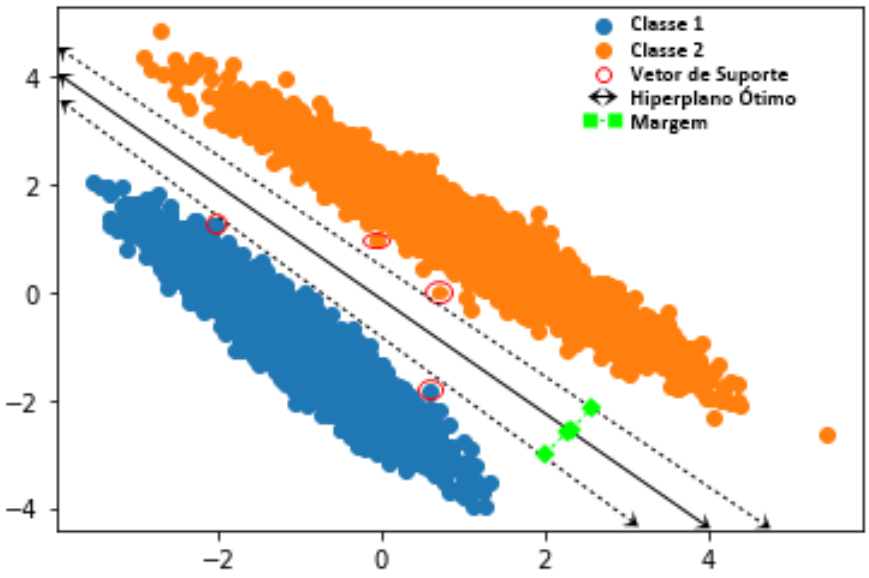
2.3 Máquina de Vetores de Suporte

Máquina de vetor de suporte é um algoritmo de aprendizado supervisionado que procura encontrar uma superfície de separação entre as classes realizando mapeamentos para outros espaços.

Os vetores criados pelo SVM são mapeados de forma não linear para um espaço de alta dimensão e definem hiperplanos que generalizam as classes. O algoritmo procura pela maior margem de separação entre os vetores das classes envolvidas. A margem é definida pelo cálculo da distância dos vetores de suporte com o hiperplano. Quanto maior a margem, melhor o hiperplano encontrado. Os vetores de suporte por sua vez, consistem nas amostras das classes mais próximas ao hiperplano avaliado (CORTES; VAPNIK, 1995).

A Figura 10 ilustra esse processo para um exemplo com duas classes linearmente separáveis. O hiperplano que separa duas classes é indicado pela reta no centro do gráfico da Figura 10.

Figura 10 – Representação de um hiperplano ótimo gerado por um SVM no plano.



Fonte: O autor.

Para um conjunto de dados linearmente separáveis ilustrados na Figura 10, denotados como $\mathbf{x}$ com os rótulos $\mathbf{y}$ definidos como 1 e -1 existe um vetor $\mathbf{W}$ de pesos e um escalar $\mathbf{b}$, que supre a inequação:

$$\begin{aligned} \mathbf{W} \cdot x_i + \mathbf{b} \geq 1 \text{ se } y_i = +1, \\ \mathbf{W} \cdot x_i + \mathbf{b} \leq -1 \text{ se } y_i = -1, \end{aligned} \tag{5}$$

em que +1 representa uma classe e -1 outra.

O hiperplano ótimo é definido pela equação:

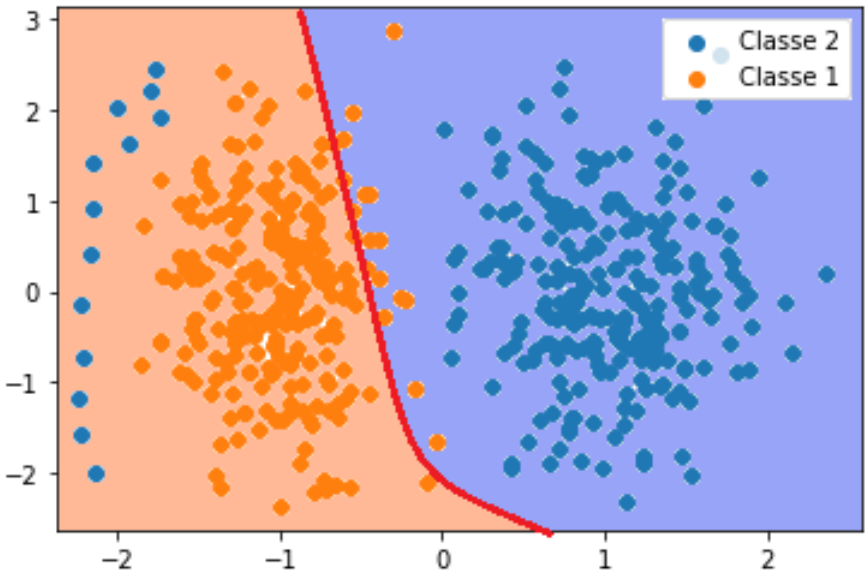
$$\mathbf{W} \cdot x + \mathbf{b} = 0. \tag{6}$$

Como mencionado, os dados mais próximos deste hiperplano são os vetores de suporte, que definem a margem.

Como nem sempre a separação entre classes é perfeita (i.e. pode ocorrer *outliers*), o algoritmo define uma margem estabelecida que aceita uma certa quantidade de erro. Para controlar este erro há dois parâmetros, C e γ, que, consecutivamente, influenciam no erro e na distância considerada para definir a superfície de decisão.

Um valor pequeno para C , flexibiliza o erro admitido no algoritmo, fazendo com que a hiperplano de decisão seja menos ajustado. Esse processo é ilustrado na Figura 11 abaixo.

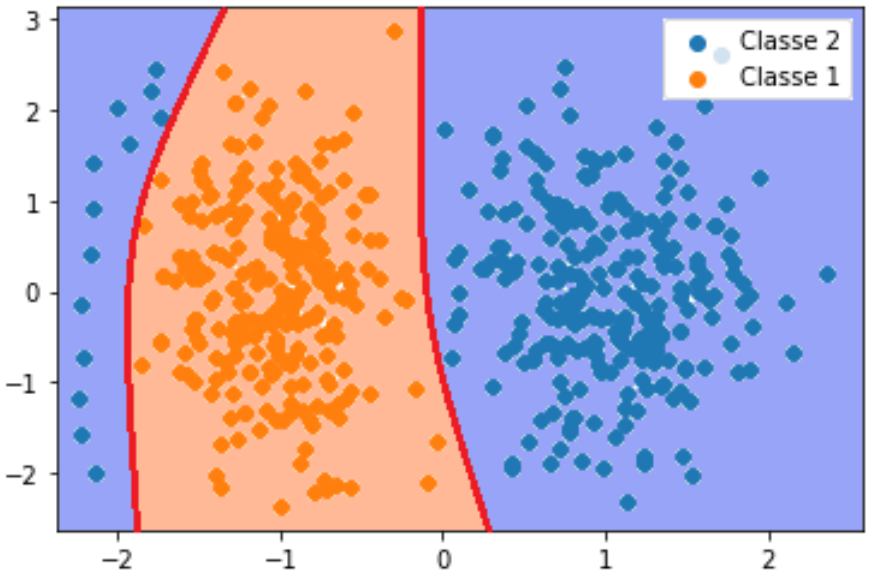
Figura 11 – Representação de um hiperplano gerado por um SVM no plano, com o parâmetro C com valor baixo.



Fonte: O autor.

Um valor maior de C leva a definição de um hiperplano que minimiza o erro (CORTES; VAPNIK, 1995). Porém, ao construir um hiperplano com o menor erro possível temos um algoritmo com maior complexidade, pois a regularização imposta pelo C, faz com que o algoritmo tenha que realizar mais interações para se ajustar.

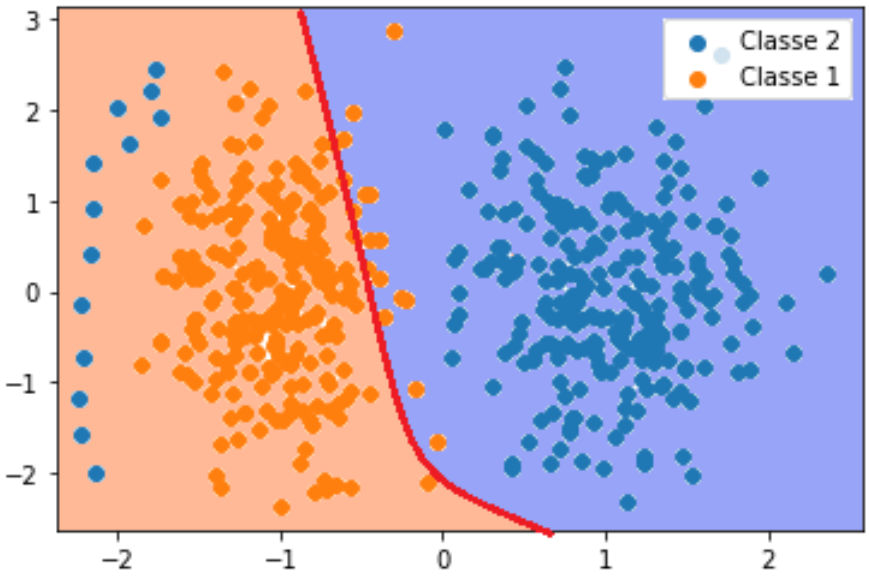
Figura 12 – Representação de um hiperplano gerado por um SVM no plano, com o parâmetro C com valor alto.



Fonte: O autor.

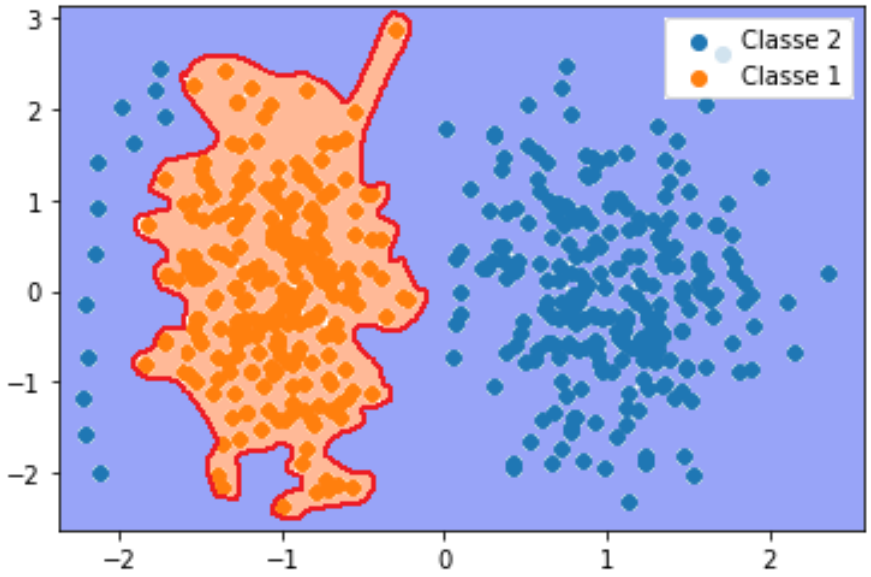
Um γ alto, segundo (AL-MEJIBLI; ALWAN; ABD, 2020) faz com que o modelo crie regiões de decisão mais exatas em torno dos dados, um valor baixo faz com que a região de decisão em torno dos dados seja maior.

Figura 13 – Representação de um hiperplano gerado por um SVM no plano, com o parâmetro γ com valor baixo.



Fonte: O autor.

Figura 14 – Representação de um hiperplano gerado por um SVM no plano, com o parâmetro γ com valor alto



Fonte: O autor.

O parâmetro γ se aplica apenas para SVMs de problemas não lineares, usados com os *kernels*, estes que segundo (AMAMI; AYED; ELLOUZE, 2015) são funções que medem similaridade entre as amostras mesmo em limites de decisão complexos, onde os mais comuns são o polinomial, o sigmóide e o RBF (Kernel de Função de Base Radial).

Construir um bom classificador, independente do método utilizado, depende muito da base de dados, quantidade de classes, quantidade de amostras e equilíbrio de exemplos de cada classes. Neste trabalho foi utilizado uma técnica de sobreamostragem de minoria sintética (SMOTE), para utilizar os métodos de classificação supracitados.

2.4 SMOTE

SMOTE (Synthetic Minority Over-sampling Technique), ou técnica de sobreamostragem de minoria sintética, é uma técnica utilizada para corrigir o desbalanceamento entre classes no treinamento de modelos preditivos. A técnica cria novas instâncias da classe minoritária, baseada em dados que estão próximos a fronteira de decisão da classe a ser sobreamostrada.

De acordo com (CHAWLA et al., 2002), o SMOTE:

> Gera exemplos sintéticos operando em espaço de características em vez de espaço de dados. A classe minoritária é superamostrada tomando cada amostra de classe minoritária e introduzindo exemplos sintéticos ao longo dos segmentos de linha que unem todos os vizinhos mais próximos da classe minoritária.

Essas novas amostras são geradas da seguinte forma:

a) Um vetor da classe minoritária é escolhido como amostra juntamente com seu vizinho mais próximo.

b) A diferença destes dois vetores é multiplicada por um número aleatório entre 0 e 1.

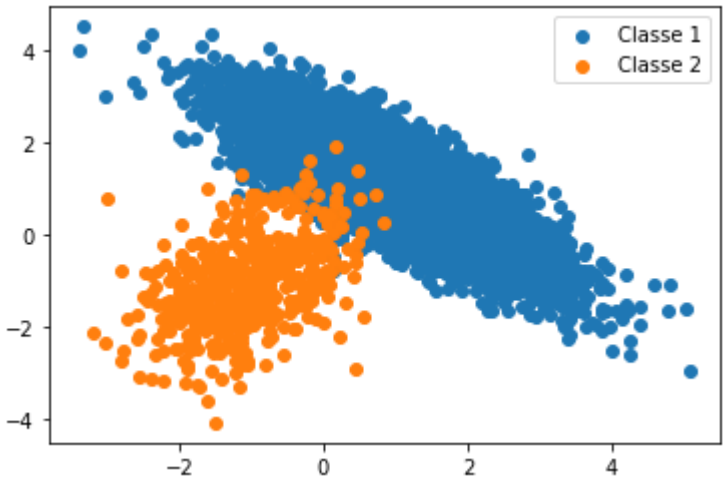
c) O resultado desta operação é adicionado ao vetor selecionado.

Essa operação faz com que seja definido um ponto entre os dois vetores da operação, a nova amostra sintética é formada nesse ponto. Desta forma, a classe minoritária é sobreamostrada até conter aproximadamente a mesma quantidade de elementos que a classe majoritária, ao mesmo tempo que força a região de decisão da classe minoritária se tornar mais geral (CHAWLA et al., 2002).

Esse procedimento da listagem é repetido criando exemplos sintéticos da classe minoritária, até que seja criado uma base com mais amostras, regiões de decisão menos específicas e equilibradas.

Exemplificando, se considerarmos um conjunto de dados desbalanceado que possui duas classes distintas como na Figura 15.

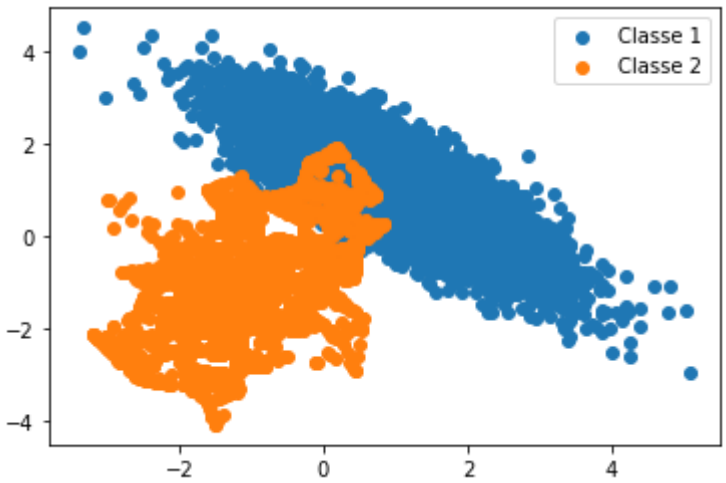
Figura 15 – Gráfico de dispersão que representa um conjunto de dados desbalanceado que possui duas classes distintas.



Fonte: O autor.

Onde a classe 2 representa apenas 5% de todos os dados, e a classe 1 95%. Após aplicar o SMOTE temos o seguinte resultado demonstrado na Figura 16.

Figura 16 – Gráfico de dispersão que representa um conjunto de dados desbalanceado após a aplicação do SMOTE.



Fonte: O autor.

Após o processo de criação de exemplos sintéticos para a classe 2, agora tem-se uma base de dados balanceada. A classe 2 possui agora 50% dos dados a serem utilizados no treinamento.

Uma desvantagem dessa técnica é que pode-se causar uma sobreposição de exemplos entre as classes na região de decisão. Para lidar com esse problema, foram criadas variações

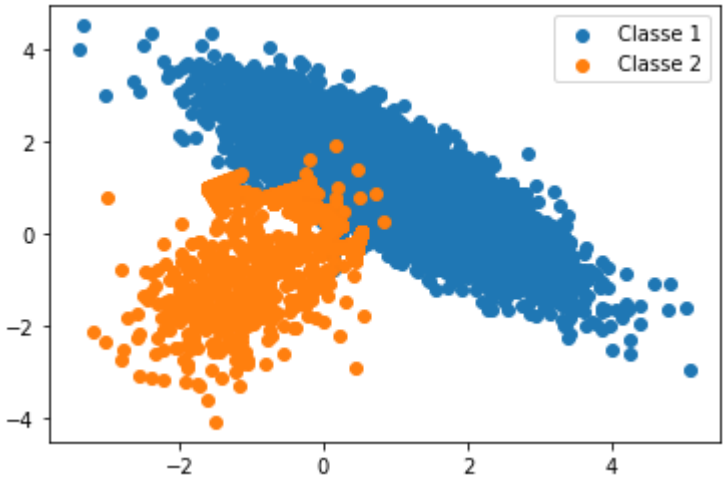
da técnica para que a sobreamostragem se torne mais seletiva ao criar elementos sintéticos, neste trabalho, utilizamos o SMOTE Borderline e o SMOTE Adasyn.

2.4.1 SMOTE Borderline

O SMOTE Borderline é uma variação do SMOTE, que explora a caraterística de que os algoritmos de classificação são destinados a aprender uma função para distinguir o limite de decisão de cada classe a ser classificada (HAN; WANG; MAO, 2005). Os dados sobreamostrados por essa variação tem como origem imagens que se encontram na fronteira de decisão entre as classes.

Exemplificando, se considerarmos a distribuição de dados desbalanceados ilustrado na Figura 15 ao aplicar o SMOTE Borderline temos distribuição plotada na Figura 17.

Figura 17 – Gráfico de dispersão que representa um conjunto de dados desbalanceado após a aplicação do SMOTE.



Fonte: O autor.

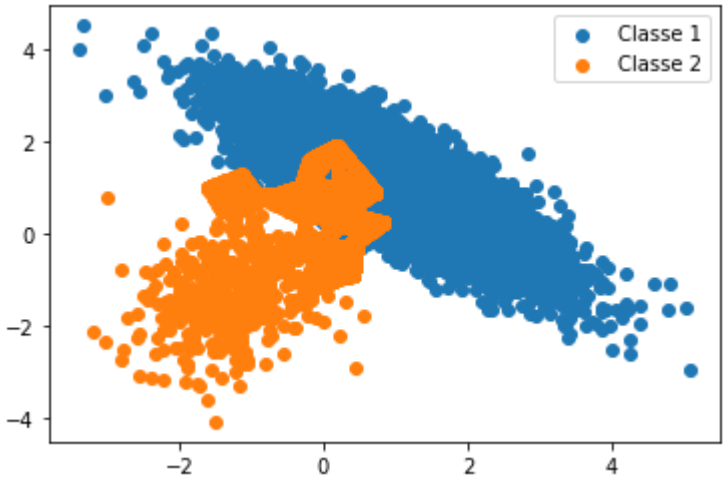
Notamos que o algoritmo focou em sobreamostrar a fronteira de decisão entre as duas classes, igualando assim a quantidade de elementos entre as classes e com diversos pontos sobrepostos na divisão entre as classes.

2.4.2 SMOTE Adasyn

O SMOTE Adasyn é uma variação do SMOTE que cria um modelo discriminativo para gerar novos exemplos sintéticos nos locais onde a densidade da classe minoritária é baixa no espaço de características (HE et al., 2008). Dessa forma, diferentemente da abordagem anterior, o número de amostras para classe minoritária é decidido durante a execução do algoritmo.

Exemplificando, se considerarmos a distribuição de dados desbalanceados ilustrado na Figura 15 ao aplicar o SMOTE Adasyn temos a seguinte distribuição:

Figura 18 – Gráfico de dispersão que representa um conjunto de dados desbalanceado após a aplicação do SMOTE.



Fonte: O autor.

Esta variação do SMOTE sobreamostrou os elementos da fronteira, porém, causou uma sobreposição maior entre as classes que o método anterior.

Capítulo 3

Materiais e Métodos

Este Capítulo apresenta a base de dados utilizada, o pré-processamento realizado e as ferramentas e métricas utilizadas no desenvolvimento e avaliação dos algoritmos de classificação.

3.1 Ferramentas Utilizadas

Para o desenvolvimento desse trabalho, utilizamos um ambiente de código aberto utilizando a linguagem Python (versão 3.7.1) que possui diversas bibliotecas disponíveis para inteligência artificial e manipulação de dados. A IDE (Integrated Development Environment) utilizada foi Spyder na versão 4.2, por ser um software amigável para o desenvolvimento em Python. As principais bibliotecas utilizadas foram:

a) Keras[1] versão 2.3.1

b) TensorFlow[2] versão 1.14.0 e 2.4.0

c) Imblearn[3] versão 0.7.0

d) Splitfolders[4] versão 0.4.3

e) Pandas[5] versão 1.1.1

f) Scikit-learn[6] versão 0.24.1.

O processamento foi realizado nos seguintes equipamentos:

a) Um notebook equipado com um processador Intel i7-7700HQ com a frequência de 2.80GHz, memória RAM DDR4 com 16GB, equipado com uma placa de vídeo Nvidia GTX 1050Ti com 4GB de VRAM;

[1] https://keras.io
[2] https://www.tensorflow.org
[3] https://imbalanced-learn.org/stable
[4] https://pypi.org/project/split-folders
[5] https://pandas.pydata.org
[6] https://scikit-learn.org
[7] https://azure.microsoft.com

b) uma máquina virtual da plataforma Azure[7] da Microsoft equipada com um processador AMD EPYC 7452 com a frequência de 2.35 GHz com 32 núcleos (nos experimentos foram utilizados apenas 20 núcleos virtuais), memoria RAM de 160 GB, sem placa dedicada de vídeo, sistema operacional Windows 10 e

c) uma máquina virtual equipada com um processador Intel Xeon E5-2690 V3 com a frequência de 2,60 GHz, com 6 núcleos, memória RAM de 56GB e placa de vídeo dedicada Nvidia Tesla K80 com 12GB de VRAM disponíveis dos 24GB e o sistema operacional Windows Server 2019.

3.2 Base de Dados

Os dados utilizados tem origem na base colaborativa e de acesso aberto ISIC (*International Skin Imaging Collaboration*), que disponibiliza anotações e imagens de lesões de pele. Esta base, que advém de uma parceria entre academias e indústria e serve para facilitar o desenvolvimento de aplicações que envolvam imagens de pele, com o objetivo de reduzir a mortalidade por cânceres de pele melanoma (ISIC, 2018).

Essa base contém (até o momento do download em fevereiro de 2021) 69.437 imagens cada qual com seu respectivo arquivo de anotações. Desse total, 57.540 das imagens representam tumores benignos e 6.194 malignos.

Os métodos de diagnóstico foram divididos em:

a) Imagem em série sem alterações.

b) Sem informações.

c) Microscopia confocal com consenso dermatológico.

d) Consenso de um especialista.

e) Histopatologia.

As imagens e anotações foram obtidas utilizando a API [8] disponibilizada pela própria ISIC. Todas as imagens foram baixadas em formato `jpeg` preservando as resoluções originais. No total, o tamanho ocupado pelas imagens e arquivos de anotações em disco foi, respectivamente, de 102 `GB` e de 270 `MB`.

3.3 Pré-processamento dos Dados

Os arquivos de anotações disponibilizados pela ISIC foram utilizados para realizar a organização e refinamento das imagens. Um algoritmo foi desenvolvido para processar os arquivos de anotações, extrair as informações sobre as imagens e criar uma tabela com o formato da Tabela 1.

[8] https://isic-archive.com/api/v1/

Tabela 1 – Formato da tabela de informações sobre as imagens da base de dados ISIC

Id	**Nome**	**Benigna ou Maligna**	**Diagnóstico**	**Tipo de Diagnóstico**	**Sexo**	**Tipo de Imagem**	**Endereço**
1	ISIC1	Maligna	Melanoma	Histopatologia	M	Dermatoscópica	C:/isic1.jpg
2	ISIC2	Benigna	Nevus	Histopatologia	F	Dermatoscópica	C:/isic2.jpg
...	...	...	...	...	...	...	...
N	ISICN	Benigna	Nevus	Histopatologia	F	Dermatoscópica	C:/isicN.jpg

Fonte: O autor.

As informações da Tabela 1 foram utilizadas em etapas posteriores desse estudo para organizar e segmentar a base de dados.

3.4 Métricas

Métricas são medidas passíveis de quantificação, que são usadas para avaliar, analisar e tomar decisões, neste trabalho foram utilizadas as seguintes métricas:

a) Média.

b) Desvio Padrão.

c) Distância Euclidiana.

d) Acurácia.

e) Função de perda (loss).

f) Curva ROC.

g) Área sob a curva (AUC)

h) Validação Cruzada.

A média é a métrica que indica um valor típico que se repete na população, ou seja informa um valor central em torno dos valores de uma determinada variável (PINHEIRO et al., 2009).

A média utilizada neste trabalho foi a média aritmética, que é calculada através da soma de todos elementos da série dividido pela quantidade total de observações (BUSSAB; MORETTIN, 2010), e pode ser calculada utilizando:

$$\mu = (x_1 + x_2 + ... + x_n)/n = \sum_{i=1}^{n} x_i \Big/ n \tag{7}$$

Onde μ é a média, x_i, os elementos do conjunto e n a quantidade de elementos.

O desvio padrão segundo (PINHEIRO et al., 2009) é um indicador de dispersão, que aponta o grau de espalhamento dos valores em torno da média.

Portanto, podemos utilizar essa métrica para definir o quão dispersos os dados de um conjunto são, (PINHEIRO et al., 2009) define o cálculo do desvio padrão utilizando:

$$S = \left(\sum_{i=0}^{n} (x_i - \mu)^2 / n - 1 \right)^{1/2}. \tag{8}$$

Onde S é o desvio padrão, x_i, os elementos do conjunto, μ é a média e n a quantidade de elementos.

Em outras palavras, segundo (BUSSAB; MORETTIN, 2010), o desvio padrão demonstra em média qual é o erro ao substituir cada observação por um valor que representa o resumo de todos os dados do conjunto.

A distância euclidiana, comumente encontrada na literatura como norma L2, é uma métrica pertencente a família LP métricas de similaridade, essas métricas são utilizadas frequentemente ao realizar operações para encontrar diferenças entre os pixeis de imagens a serem comparadas (SINHA; RUSSELL, 2011).

E para encontrar a diferença entre os vetores que representam as imagens, (SINHA; RUSSELL, 2011) define que distância euclidiana ou norma L2 é calculada com:

$$E(x, y) = \left(\sum_{i=0}^{n-1} (x_i - y_i)^2 \right)^{1/2}, \tag{9}$$

em que $E(x, y)$ indica a distância euclidiana em relação aos vetores x e y, x_i e y_i os elementos de cada vetor e n a quantidade de elementos dos vetores, que neste caso representam as imagens.

A acurácia é uma métrica que visa refletir a taxa de acerto global do classificador (DEVELOPERS, 2018) e é definida por:

$$Acurácia = \frac{Número\ de\ previsões\ corretas}{Número\ de\ previsões\ incorretas}. \tag{10}$$

Note que o valor é um número entre 0 e 1. Quanto maior o valor da acurácia (mais próximo de 1), melhor é a performance do modelo avaliado. Quando há informações sobre os tipos de erros envolvidos (α e β), utiliza-se a seguinte definição (DEVELOPERS, 2018):

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN}, \tag{11}$$

em que VP corresponde ao número de casos verdadeiros positivos, VN de verdadeiros negativos, FP de falsos positivos e FN de falsos negativos. Verdadeiro positivo e verdadeiro negativo referem-se às previsões corretas geradas pelo modelo (respectivamente, previsões de câncer e tumor benigno). Falso positivo e falso negativo estão relacionadas com os erros que acontecem, respectivamente, quando prevê que uma lesão benigna é um câncer ou que um câncer é uma lesão benigna.

A função de perda ou *loss* é a função responsável por indicar e penalizar previsões ruins (DEVELOPERS, 2018). Um bom modelo consegue minimizar esta função durante

o treinamento e por isso suas previsões estão o mais próximas da realidade. Existem diversas maneiras de calcular a perca de um modelo, neste trabalho foram utilizadas as funções de perca *categorical crossentropy* e *hinge*.

A *loss categorical crossentropy* é uma função que pode ser usada para classificação de 2 a n classes. Segundo (PELTARION, 2020) esta função foi projetada para mensurar a diferença entre duas distribuições de probabilidade e pode ser definida como:

$$\text{categorical crossentropy} = -\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i), \tag{12}$$

em que $\hat{y}_i$ corresponde ao i-ésimo valor de saída do modelo avaliado, y_i é a classe correspondente e N é a quantidade de valores de saídas do modelo (PELTARION, 2020).

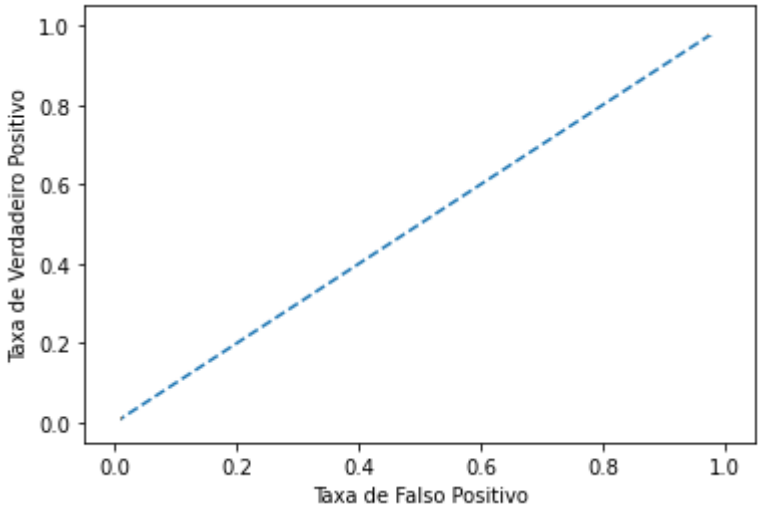
A *Hinge loss*, amplamente utilizada em funções SVM, é dado por (RENNIE; SREBRO, 2005):

$$\text{Hinge} = Max\,(0, 1 - z) = \begin{cases} 0, & \text{se } z \geq 1 \\ 1 - z, & \text{se } z < 1, \end{cases} \tag{13}$$

em que z indica a saída do classificador. Caso este valor seja maior que 1, a classificação esta correta e temos erro igual a zero. Caso contrário, temos um valor de erro que cresce a medida que o valor de z decresce.

A curva ROC (*Receiver Operating Characteristic*) é uma métrica gráfica usada para avaliar classificadores (FAWCETT, 2006). Cada curva contém gráficos bidimensionais que medem a performance do modelo avaliando a relação entre a taxa de verdadeiros positivos (representada pelo eixo y) e a taxa de falsos positivos no eixo x.

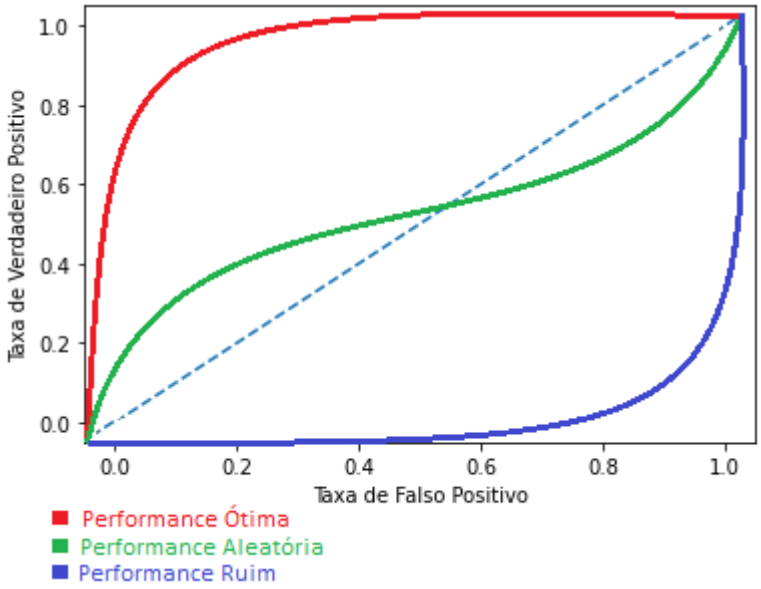
Figura 19 – Plano ROC.



Fonte: O autor.

A Figura 19 apresenta o plano em que a curva ROC é traçada. Os valores dos eixos x e y variam entre 0 e 1 que representam, respectivamente, a taxa de falsos positivos e a taxa de verdadeiro positivos. Quanto mais rápido a curva se aproximar do valor 1 melhor é o classificador. A linha tracejada no centro da Figura 19 descreve o comportamento do pior classificador binário (com respostas aleatórias).

Figura 20 – Exemplos de curvas ROC.



Fonte: O autor.

A Figura 20 apresenta um exemplo de curva em que é possível comparar o desempenho de 3 (três) classificadores. A linha vermelha descreve um modelo com boa taxa de classificação, a linha verde um modelo que adivinha rótulos de forma aleatória e a linha roxa a de um modelo que erra a classificação dos dados. Ao traçar a curva ROC é calculado a AUC, que é a área sob a curva (integral da área inferior da curva traçada no plano ROC)(DEVELOPERS, 2018).

A AUC representa a probabilidade de um modelo classificar um dado novo em sua classe correta (DEVELOPERS, 2018) e indica o quanto o modelo consegue separar dados novos em suas devidas classes. Quanto maior AUC melhor é o classificador.

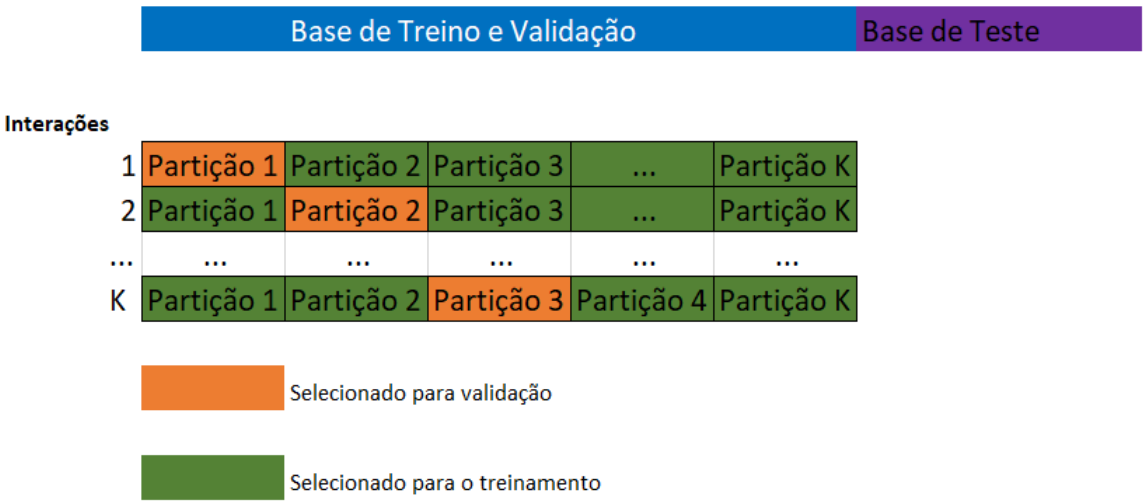
Para terminar essa seção, descrevemos brevemente o processo de avaliação de modelos utilizado em todos os experimentos deste trabalho e que é conhecido como validação cruzada. Na Validação cruzada, são definidos conjuntos de treinamento e teste independentes que se cruzam em rodadas consecutivas, evitando análises tendenciosas (REFAEILZADEH; TANG; LIU, 2009).

Mais especificamente, esse processo é realizado da seguinte maneira: os dados são divididos em k partições de tamanho aproximado, durante k iterações de treinamento e

validação, uma dessas k partições é selecionada para validação, as demais são utilizadas no treinamento (REFAEILZADEH; TANG; LIU, 2009).

A Figura 21 ilustra o funcionamento da validação cruzada, onde a cada iteração do modelo uma partição é selecionada para a validação e as demais para o treinamento, desta forma todos os dados são avaliados, os dados da base de teste são deixados de fora deste particionamento.

Figura 21 – Exemplificação da partição e funcionamento da validação cruzada.



Fonte: O autor.

As métricas média, desvio padrão e distância euclidiana foram utilizadas durante o processamento e análise da base de dados. A acurácia, função de perda, curva ROC, AUC e validação cruzada foram utilizadas para o treinamento e avaliação dos modelos obtidos durante os experimentos realizados (seção 5.3).

Capítulo 4

Análise das Imagens

Após o pré-processamento das anotações, as imagens com anotações de diagnóstico utilizando o método sem informações e por imagem em série sem alterações foram removidas da base, para que o modelo utilize apenas imagens classificadas por biópsia ou consenso médico, assim, aumentando a confiabilidade da classificação das imagens como benignas e malignas. Também foram removidas as duplicatas indicadas no desafio ISIC 2020 (ROTEMBERG et al., 2021).

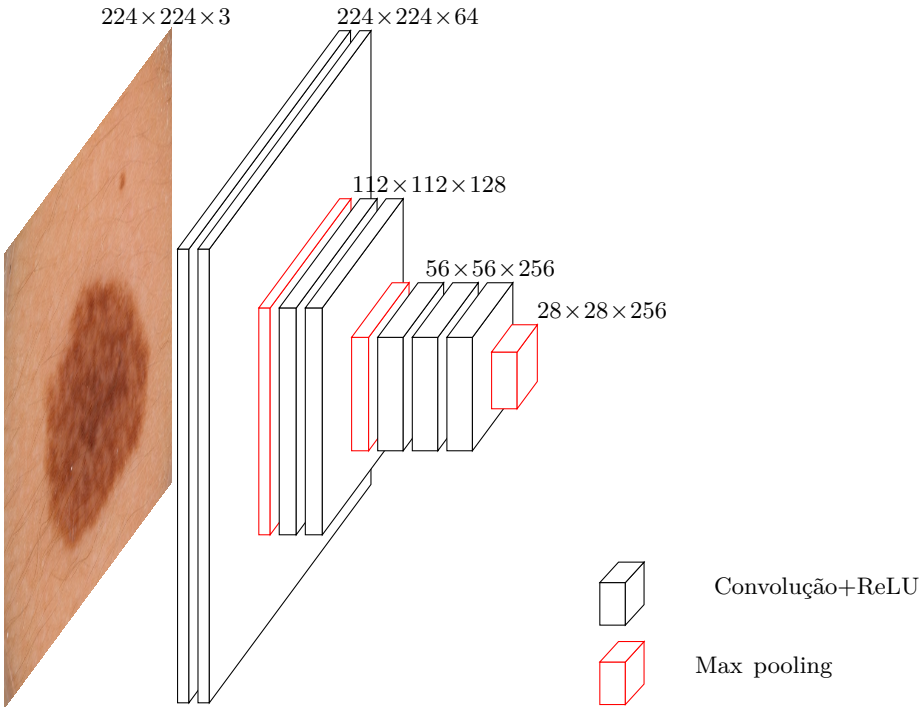
Após este processamento a base totalizou 35.457 imagens, sendo 29.279 imagens benignas e 6.178 imagens malignas. Durante a análise visual da base de dados, foi identificado imagens que continham curativos ou marcações que obstruíam as lesões, estas imagens também foram removidas. Após esta alteração a base totalizou 32.519 imagens, sendo 26.341 imagens benignas e 6.178 malignas.

Em (ESTEVA et al., 2017), as imagens semelhantes foram removidas da base de teste e validação (utilizando uma variação do algoritmo KNN), porém, utilizadas no treinamento, neste trabalho o objetivo foi remover toda e qualquer imagem semelhante, para isto cada imagem da base foi comparada com as demais imagens da mesma classe através da distância euclidiana.

Para gerar os vetores e calcular a distância euclidiana foi utilizado como descritor as dez primeiras camadas da arquitetura VGG16 (Figura 22), para que a matriz que representa a imagem fosse filtrada e mantivesse os principais recursos, este processo foi baseado em (ZHANG et al., 2018), onde diversos experimentos foram realizados comparando o desempenho da percepção de semelhança entre humanos, redes neurais profundas e as métricas mais usuais para medir semelhança entre imagens. Onde as redes neurais se destacaram em se aproximar ao nível humano de percepção de semelhança, uma das arquiteturas utilizadas no estudo foi a VGG treinada com a base ImageNet [9] e por este motivo a VGG16 (SIMONYAN; ZISSERMAN, 2015) foi escolhida para este estudo.

[9] http://www.image-net.org

Figura 22 – Representação das camadas da arquitetura VGG16 utilizada.



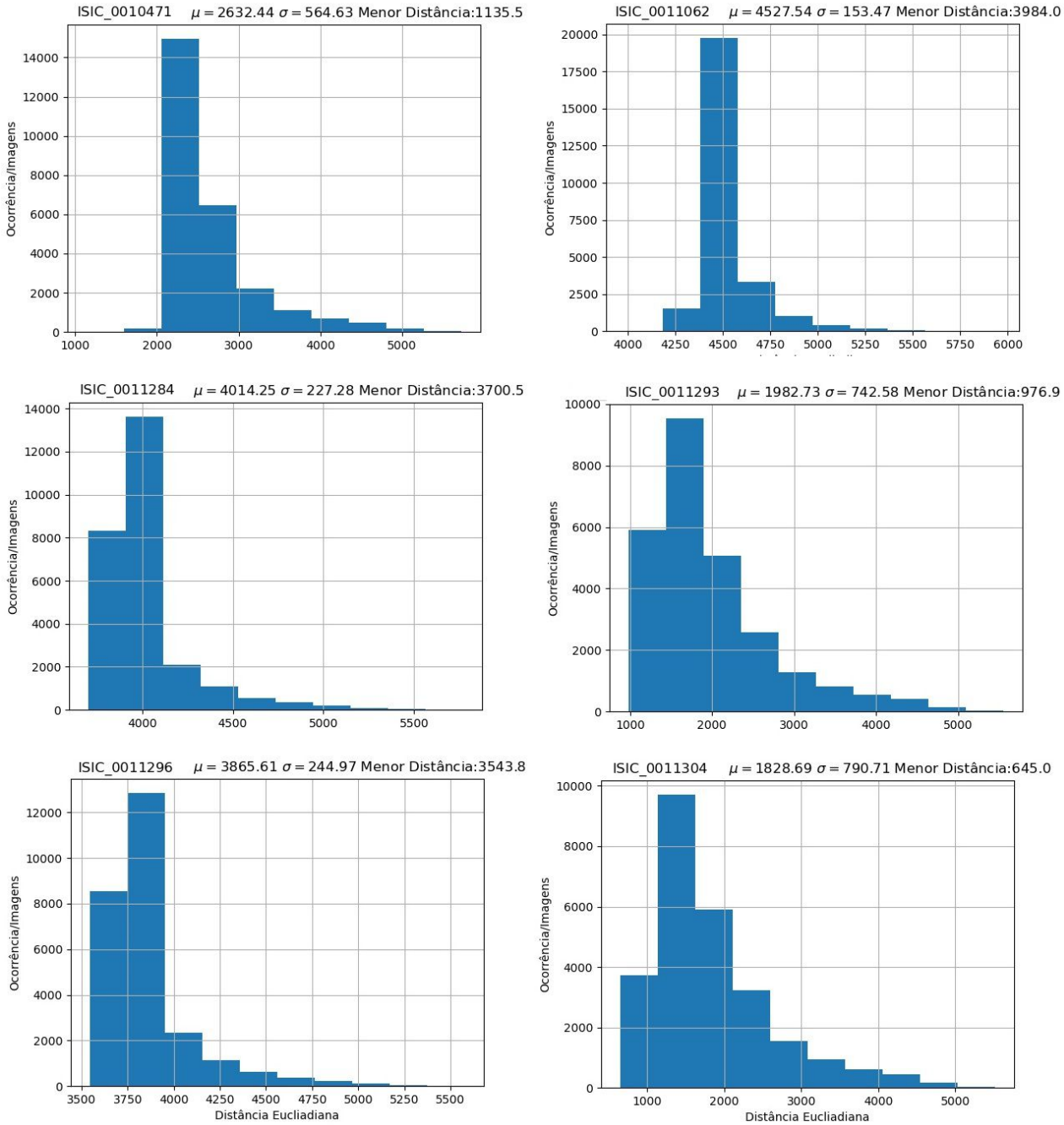
Fonte: O autor.

A saída deste processamento é uma matriz de dimensão $(28 \times 28 \times 256)$. Para facilitar a comparação, cada matriz foi redimensionada para um vetor de tamanho $1 \times 1 \times 200.704$. Cada vetor foi comparado com os demais vetores de sua classe e um histórico de distância foi criado. Este histórico pode ser visualizado em histogramas com a distribuição da distância de cada imagem da base de dados. Além desse histograma, foi avaliada a média das distâncias em relação a imagem, o desvio padrão e a menor distância encontrada.

A partir da análise desses dados, para evitar problemas no treinamento, foram removidas da base de dados as imagens com distância igual a zero em relação a imagem comparada. Uma vez removida, não eram consideradas nas próximas comparações.

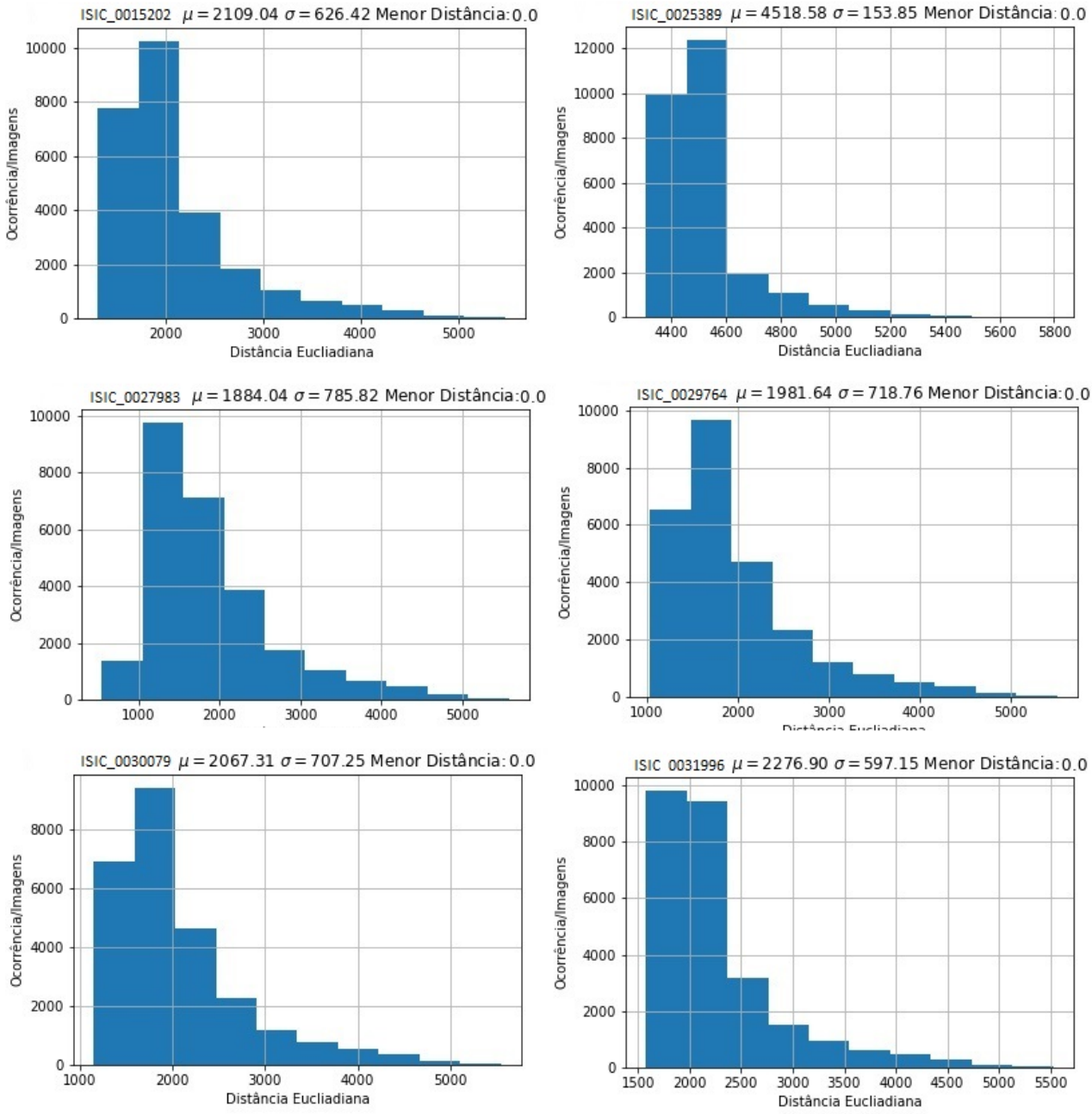
A decisão de remoção foi tomada levando em consideração que a média entre as imagens sempre estava no intervalo de 1.000 a 5.000, o desvio padrão entre 200 a 1.000. Durante a inspeção visual dos histogramas foi calculado também o coeficiente de variação que sempre estava em torno de 30%. Estas métricas indicaram que as imagens são bastante diversas no geral. As Figuras 23 e 24 mostram os histogramas resultantes com as comparações realizadas, incluindo as imagens removidas (Figura 24).

Figura 23 – Exemplos dos histogramas gerados durante o processamento das imagens.



Fonte: O autor.

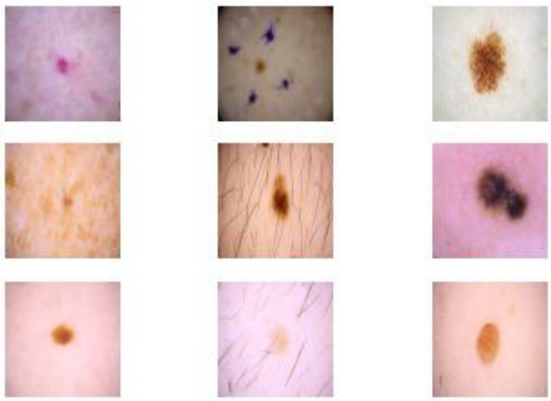
Figura 24 – Exemplos dos histogramas das imagens removidas gerados durante o processamento das imagens.



Fonte: O autor.

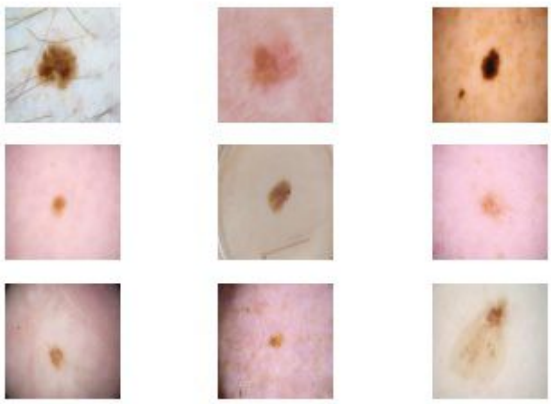
O histórico também foi utilizado para gerar quadros com as imagens mais semelhantes. As Figuras 25 e 26 mostram alguns exemplos de imagens semelhantes, considerando as métricas utilizadas no trabalho.

Figura 25 – Exemplo dos quadros das imagens mais semelhantes encontrados durante o processamento.



Fonte: O autor.

Figura 26 – Exemplo dos quadros das imagens mais semelhantes encontrados durante o processamento.

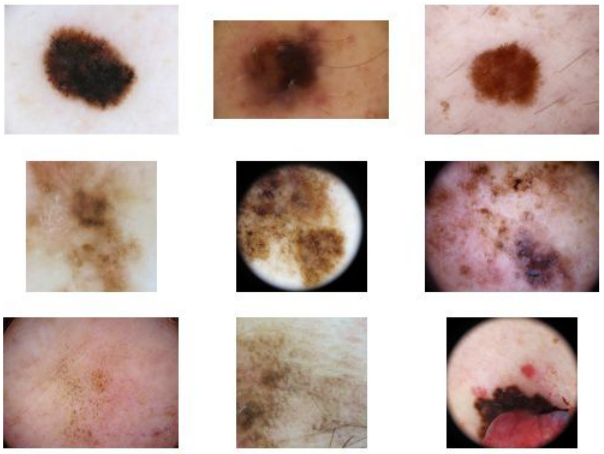


Fonte: O autor.

Os quadros apresentados nas Figuras 25 e 26, mostram as 8 (oito) imagens mais próximas em relação a primeira imagem no canto superior esquerdo.

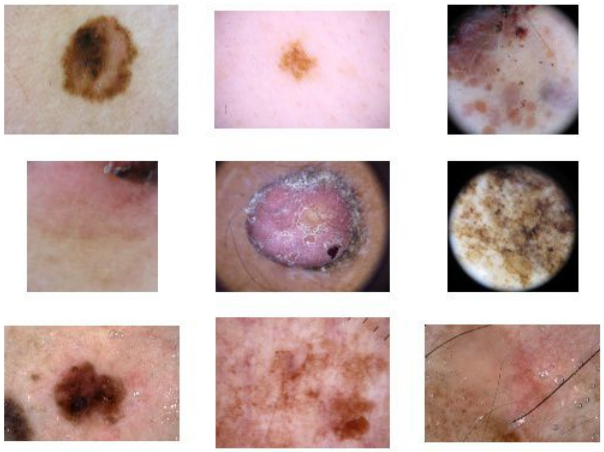
Seguindo a mesma lógica dos quadros acima, as Figuras 27 e 28 mostram as imagens mais distantes encontradas na base de dados. Novamente, o processo foi calculado considerando a imagem no canto superior esquerdo.

Figura 27 – Exemplo dos quadros das imagens mais dissemelhantes encontrados durante o processamento.



Fonte: O autor.

Figura 28 – Exemplo dos quadros das imagens mais dissemelhantes encontrados durante o processamento.



Fonte: O autor.

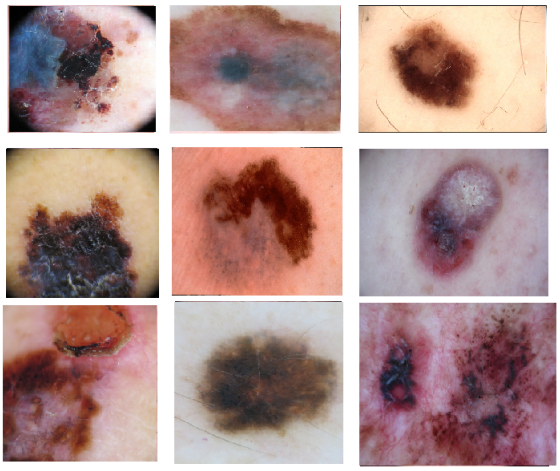
Após a remoção das imagens muito semelhantes, a base de dados ficou com 32.466 imagens, sendo 26.291 benignas e 6.175 malignas. Há um desbalanceamento significativo entre as classes, pois as imagens de lesões benignas representam 80,98% das imagens. Este desbalanceamento pode comprometer o aprendizado do modelo (SOMASUNDARAM; REDDY, 2016), já que o aprendizado depende da distribuição do conjunto de dados. Poucas amostras fazem com que o classificador seja treinado de forma insuficiente para as classes minoritárias, chegando a ponto de poderem ser ignoradas.

Para resolver este problema utilizamos as técnicas da seção 2.4, apenas nos dados de treinamento. Para isto, a base passou por um processo de separação, onde foi separada em imagens para treinamento, teste e validação de forma aleatória.

A base de treino foi formada inicialmente por 22.725 imagens, sendo 18.403 para tumores benignos e 4.322 para malignos. Após isto, foi aplicado o algoritmo SMOTE Bordeline e SMOTE Adasyn na base de treinamento. A sobreamostragem resultante da aplicação do SMOTE Bordeline e SMOTE Adasyn devolveu, respectivamente, um conjunto de 18.403 e 18.176 novas imagens malignas. Com isso, foram formadas 2 (duas) bases de treinamento com, respectivamente, 41.128 e 40.901 imagens cada.

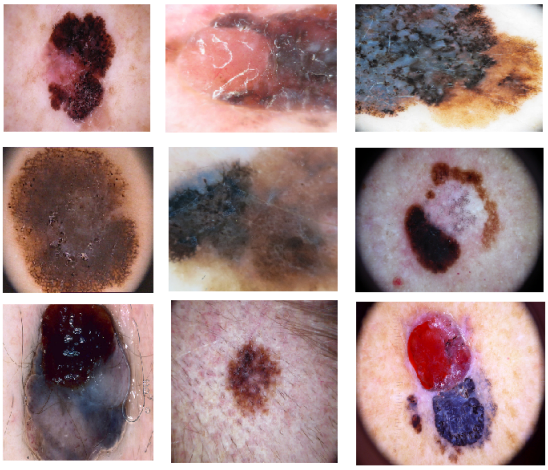
As Figuras 29 e 30 mostram as imagens geradas pelos algoritmos de reamostragem SMOTE Adasyn e Bordeline, respectivamente. Como pode ser observado, as imagens geradas tem características similares as imagens originais da base.

Figura 29 – Exemplo de imagens geradas pelo algoritmo de reamostragem SMOTE Adasyn.



Fonte: O autor.

Figura 30 – Exemplo de imagens geradas pelo algoritmo de reamostragem SMOTE Borderline.



Fonte: O autor.

As bases de validação foram readequadas, para que a quantidade de imagens de cada classe contivessem, aproximadamente, a mesma quantidade. As imagens benignas removidas, foram adicionadas a base de treinamento. Ao final desta adequação, a base de treino reamostrada pelo SMOTE Borderline ficou com 43.819 imagens sendo 21.170 benignas e 22.649 malignas, a base reamostrada pelo SMOTE Adasyn com 43.674 imagens sendo 21.170 imagens benignas e 22.504 malignas.

Capítulo 5

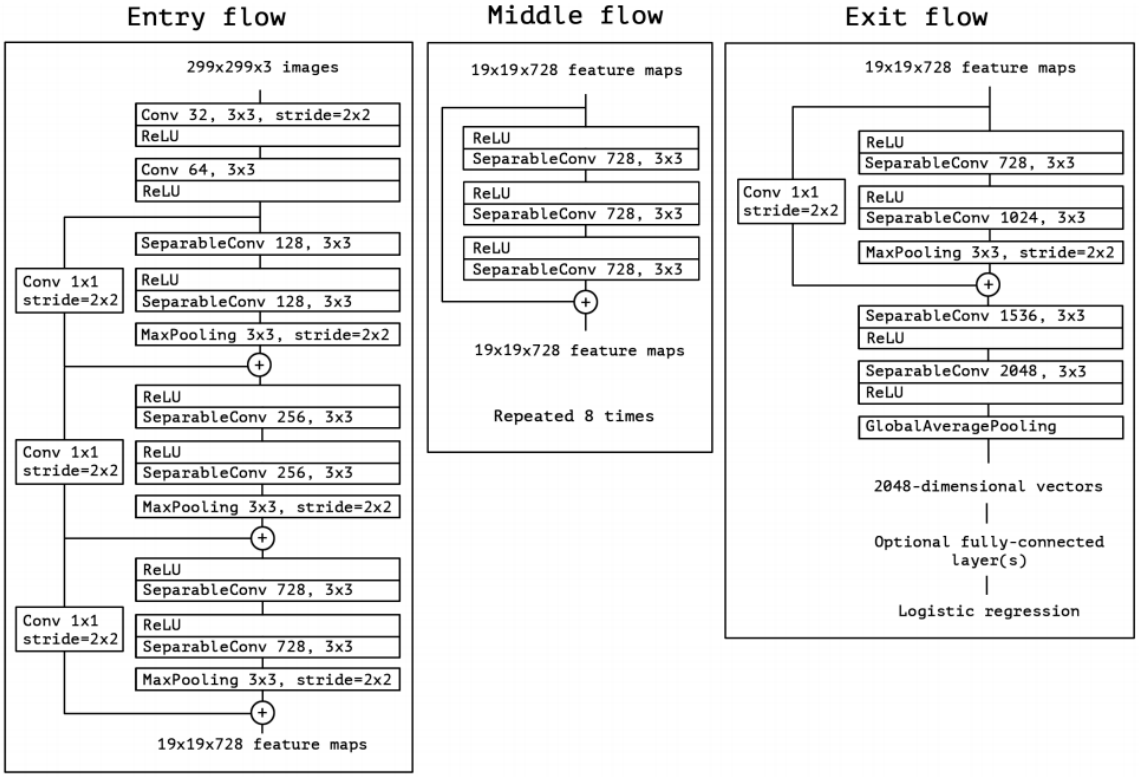
Algoritmos de Classificação

O objetivo principal deste trabalho é classificar as imagens de lesões de pele em duas classes que representam tumores benignos e malignos. Essa classificação foi realizada utilizando as arquiteturas de redes convolucionais consideradas estado da arte na literatura **Xception** (CHOLLET, 2017) e **InceptionResNetV2** (SZEGEDY et al., 2016). Os detalhes dessas arquiteturas são apresentados a seguir.

5.1 Xcpetion

A rede Xcpetion (CHOLLET, 2017) é baseada na IncpetionV3 (SZEGEDY et al., 2015) e ambas tem quase o mesmo número de parâmetros. A Figura 31 mostra uma representação visual da arquitetura. Para facilitar o entendimento, é dividida em 3 (três) fluxos: entrada, intermediários e de saída.

Figura 31 – Representação da arquitetura Xcpetion



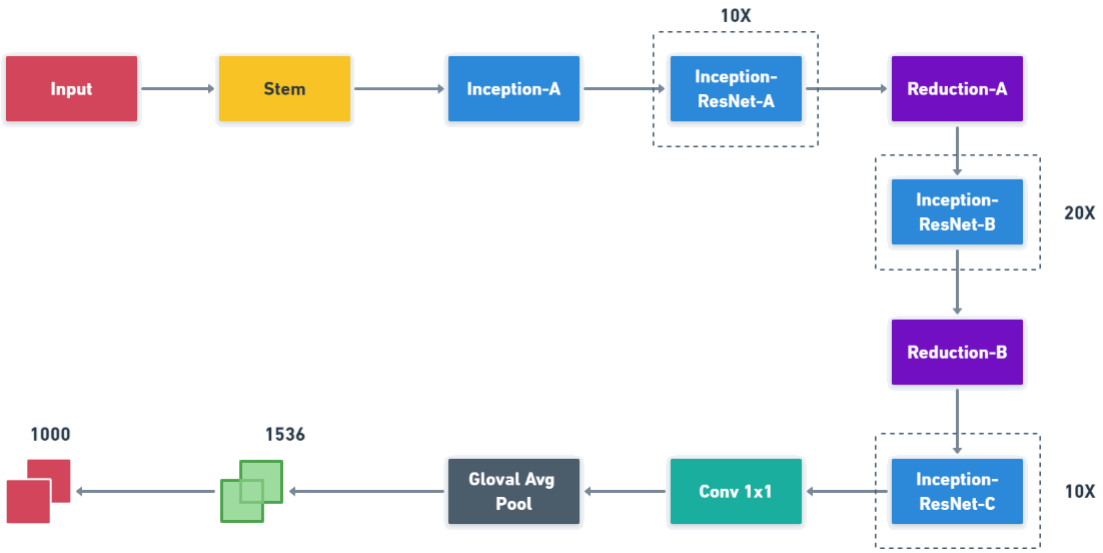
Fonte: (CHOLLET, 2017)

Como pode ser observado na Figura 31, inicialmente, os dados (i.e. imagens de dimensão $299 \times 299 \times 3$) passam uma vez pelo fluxo de entrada. Logo após, se inicia um conjunto de convoluções combinadas e o resultado dessas convoluções passa pelo fluxo do intermediário, que é repetido 8 vezes. A saída deste processo passa pelo fluxo de saída, cujas últimas camadas são totalmente conectadas. Neste trabalho, removemos as camadas totalmente conectadas da rede e, portanto, a Xception utilizada termina na camada de Global Average Pooling, cuja dimensão é 1×2048.

5.2 InceptionResNetV2

Tal como a arquitetura anterior, a InceptionResNetV2 também é uma evolução da IncpetionV3, esta variação da rede foi desenvolvida e publicada por (SZEGEDY et al., 2016). A Figura 32 mostra uma representação visual de sua arquitetura.

Figura 32 – Representação resumida da arquitetura da rede InceptionResNetV2

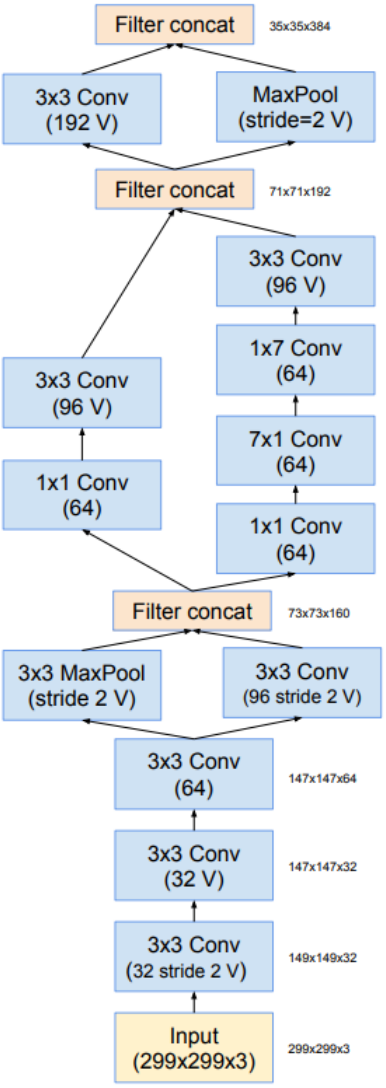


Fonte: Adaptado de (ALEMI, 2016)

Como pode ser observado na Figura 32, a arquitetura é composta pelos seguintes componentes: o bloco Stem representado pela Figura 33, Inception-A pela Figura 34, Inception-ResNet-A Figura 35, Reduction-A Figura 36, Inception-ResNet-B Figura 37, Reduction-B Figura 38 e por fim Inception-ResNet-C pela Figura 39.

O bloco Stem é o bloco responsável pelo processamento da entrada desta rede, ele recebe dados (i.e imagens) com a dimensão $299 \times 299 \times 3$ e inicia as convoluções combinadas, a saída deste bloco consiste na concatenação de uma convolução e um Max Pooling que segue para o bloco Inception-A.

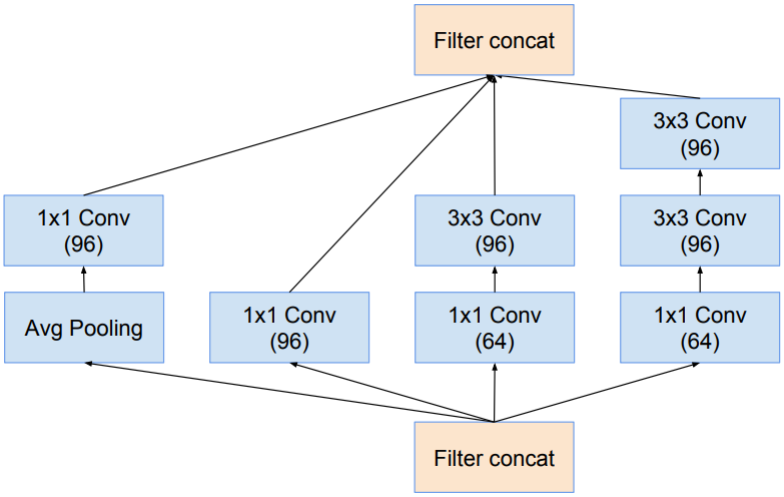
Figura 33 – Representação do bloco Stem.



Fonte: (SZEGEDY et al., 2016)

O bloco Inception-A, recebe a saída do Stem e se divide em 4 (quatro) sub redes onde as convoluções e pooling são realizados de forma independentes e são concatenados apenas na saída que segue para o bloco Inception-ResNet-A.

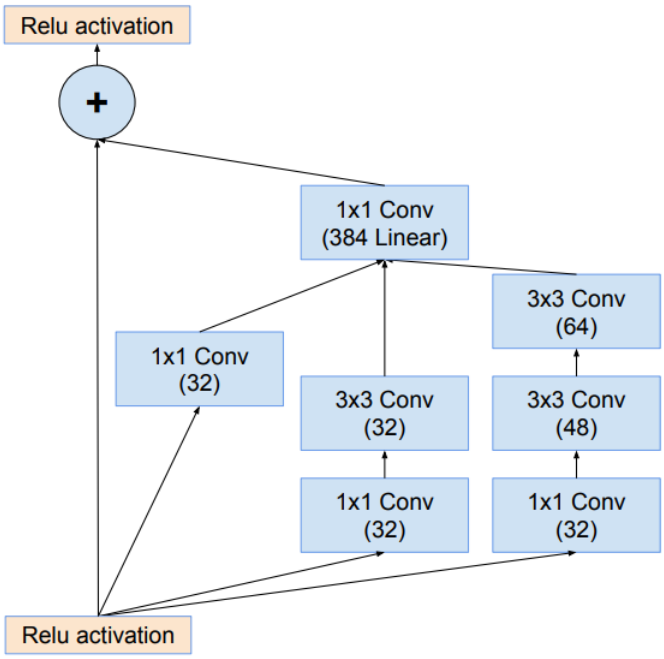
Figura 34 – Representação do bloco Inception-A.



Fonte: (SZEGEDY et al., 2016)

O bloco Inception-ResNet-A recebe a entrada do bloco anterior e aplica a função de ativação ReLu, após isto o resultado desta operação é mantido e uma cópia passa por convoluções combinadas, o resultado das convoluções combinadas é redimensionado para que as dimensões coincidam com a mantida após a aplicação da ReLu, que são combinadas e passam para o próximo bloco. Este bloco é repetido dez vezes de forma consecutiva, posteriormente, a saída é enviada para o bloco Reduction-A.

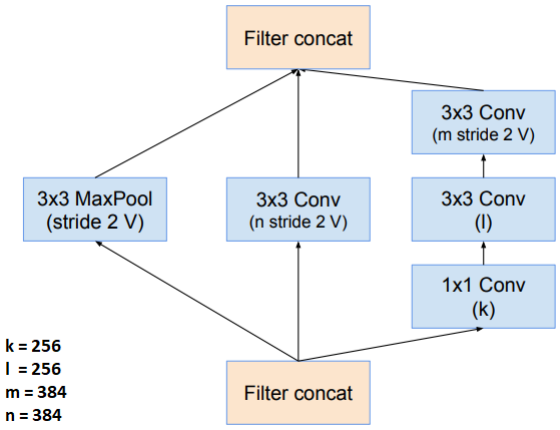
Figura 35 – Representação do bloco Inception-ResNet-A.



Fonte: (SZEGEDY et al., 2016)

O bloco Reduction-A em particular para a InceptionResNetV2 realiza as convoluções e poolings com os parâmetros $k = 256$, $l = 256$, $m = 384$ e $n = 384$, como indicado na Figura 36, após isto a saída é enviada para o bloco Inception-ResNet-B.

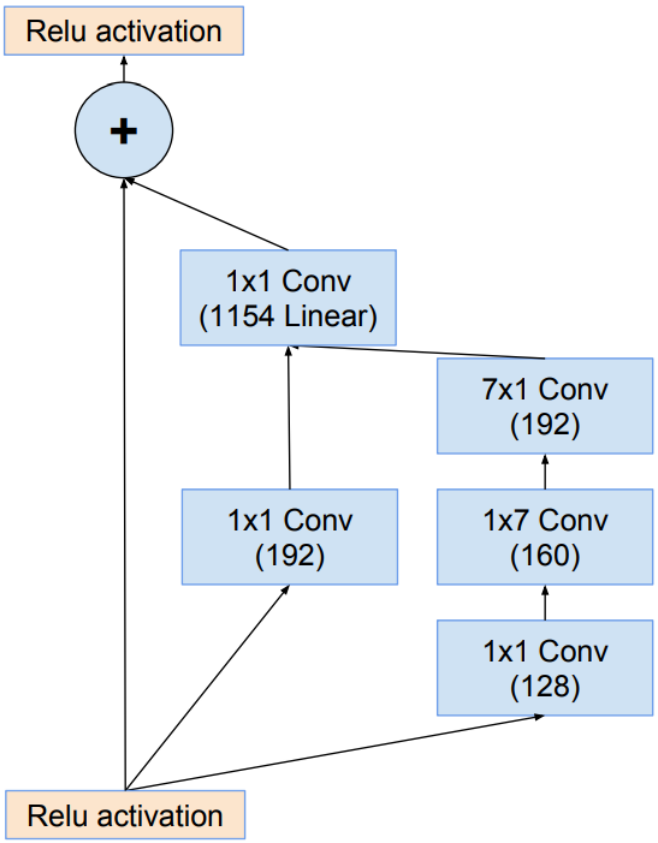
Figura 36 – Representação do bloco Reduction-A



Fonte: Adaptado de (SZEGEDY et al., 2016)

O Inception-ResNet-B, segue a mesma lógica utilizada no bloco Inception-ResNet-A, a diferença é que contém uma ramificação a menos nas convoluções combinadas. Este bloco é repetido 20 vezes e a saída final segue para o bloco Reduction-B.

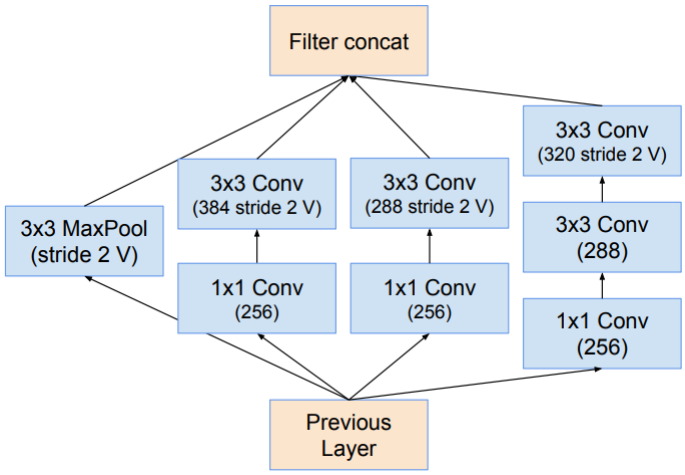
Figura 37 – Representação do bloco Inception-ResNet-B



Fonte: (SZEGEDY et al., 2016)

O bloco Reduction-B, segue o mesmo raciocínio do Reduction-A, porém, contém uma ramificação a mais de convolução na sub rede e com mais operações de convoluções nas ramificações. A saída segue para o bloco Inception-ResNet-C.

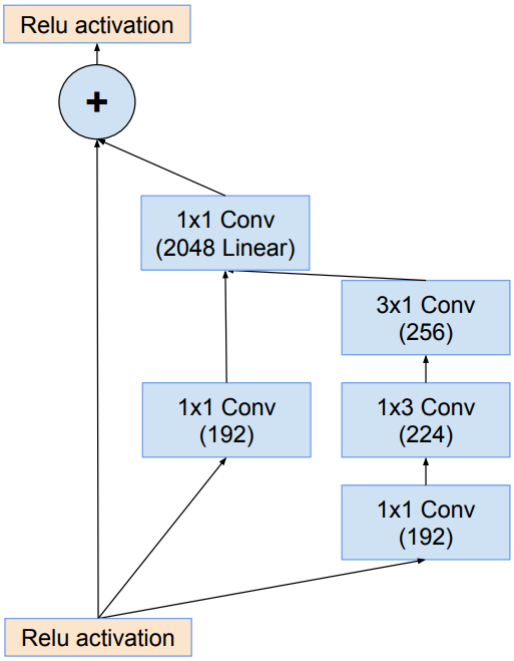
Figura 38 – Representação do bloco Reduction-B



Fonte: (SZEGEDY et al., 2016)

O bloco Inception-ResNet-C mantém o fundamento apresentado nos blocos Inception-ResNet-A e Inception-ResNet-B, com a diferença nos parâmetros na convolução. A saída deste bloco passa por uma convolução 1×1 e depois por uma Global Average Pooling que segue para a camada totalmente conectada.

Figura 39 – Representação do bloco Inception-ResNet-C



Fonte: (SZEGEDY et al., 2016)

Neste trabalho removemos a camada totalmente conectada, portanto, a InceptionResNetV2 utilizada termina no bloco de Global Average Pooling, cuja a dimensão é $1x1536$.

5.3 Detalhes do Treinamento e Validação

Com os modelos definidos, foram realizados dois grupos de treinamento utilizando a técnica denominada transferência de aprendizagem (STANFORD, 2015). No primeiro grupo, incluímos nas arquiteturas uma camada *maxpolling global* e uma função *softmax* para realizar a classificação. Já no segundo grupo, mantivemos os pesos inalterados e incluímos uma função SVM (veja Seção 2.3) para realizar a classificação.

O processo de ajuste fino consiste em uma estratégia de treinar um classificador utilizando uma arquitetura e pesos pré-existentes para um novo conjunto de dados. A ideia é treinar apenas partes do modelo, aproveitando o que foi apreendido anteriormente (STANFORD, 2015).

Para evitar o sobreajuste do modelo na nova base, apenas algumas camadas da parte superior das arquiteturas foram treinadas/alteradas neste trabalho. A maioria das cama-

das anteriores foram mantidas *congeladas* a fim de manter as características genéricas que essas redes já aprenderam em suas bases originais.

As redes Xception e InceptionResNetV2 foram treinadas, respectivamente, a partir da 126ª e da 700ª camada. As camadas anteriores, foram congeladas e os pesos de seus treinamentos originais foram utilizados.

Tal como comentamos, no primeiro grupo de treinamento, as camadas totalmente conectadas originais de cada rede foram substituídas por uma camada formada por um *maxpooling global* seguido de dois neurônios com a função de ativação *softmax*. O ajuste fino foi realizado com ambas as bases de treinamento geradas pelas técnicas de balanceamento (veja seção 2.4), e no total, foram feitos 4 treinamentos com todas as imagens normalizadas. Utilizamos *batch size* igual a 32 e a função de otimização Adam [10] com a taxa de aprendizado inicial igual a 1×10^{-4}. O treinamento foi realizado durante 50 épocas. A função *loss* foi continuamente avaliada e o treinamento era interrompido se nas últimas 5 épocas não ocorresse melhora. Para definir a parametrização da quantidade de camadas congeladas, a nova camada totalmente conectada, função de perda, batch size e demais parâmetros, foi realizado pequenos testes com uma base com 100 imagens de cada classe selecionadas aleatoriamente.

No segundo grupo de treinamento, nenhuma camada dos modelos originais foi treinada. Como todas as camadas permaneceram congeladas e as camadas totalmente conectadas originais foram removidas, este grupo de experimentos pode ser visto como a utilização das redes com vetores de características que servem de entrada para o classificador SVM.

Como a SVM leva um tempo de treinamento elevado para grande quantidade de dados, primeiro foi realizado um treinamento utilizando um SVM Linear, implementado usando a função LinearSVC da biblioteca scikit-learn. Foi utilizado C igual a 1 e a função de loss denominada Hinge [11]. Esta configuração foi escolhida para menor tempo de treinamento para bases de dados grandes.

[10] https://keras.io/api/optimizers/adam/
[11] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.hinge_loss.html

Capítulo 6

Resultados

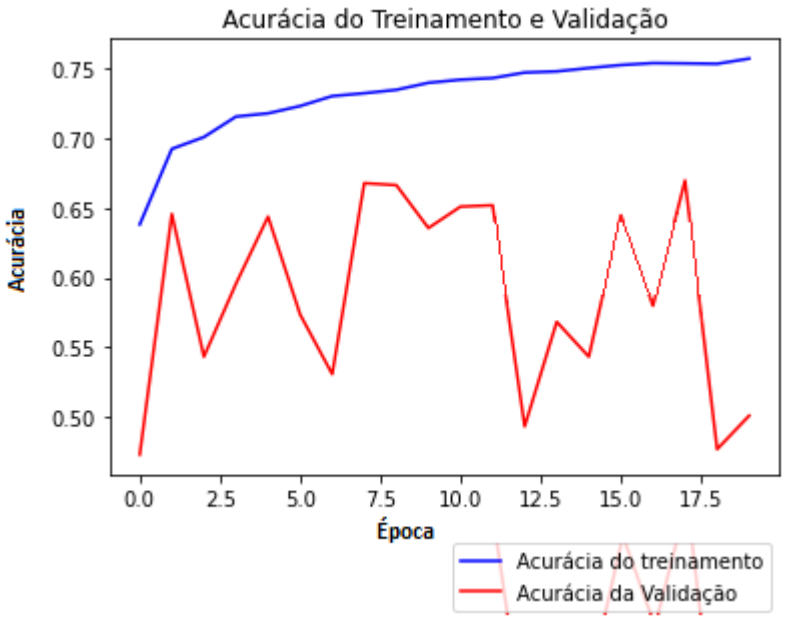
Este capítulo apresenta os resultados obtidos com as arquiteturas e esquema de testes apresentados anteriormente. Os testes realizados foram agrupados em dois experimentos, apresentados a seguir.

6.1 Experimento I (Ajuste Fino)

Os resultados apresentados nesta seção avaliam a utilização da transferência de aprendizagem com as redes Xception e InceptionResNetV2. Neste experimento, as camadas totalmente conectadas originais de ambas as redes foram substituídas por um maxpooling global seguido de dois neurônios com a função de ativação softmax. A função *loss* utilizada foi a categorical crossentropy. A *loss* foi avaliada a cada época e o treinamento foi interrompido se o valor não foi alterado após 5 (cinco) épocas. A taxa de aprendizado inicial utilizada foi de 1×10^{-4}, o batch size foi definido como 32 e a função de otimização utilizada foi a Adam. O treinamento foi realizado em ambas as bases geradas pelos algoritmos de balanceamento de base de dados.

A Figura 40 mostra a evolução da acurácia durante o treinamento e validação com rede Xcpetion e a base reamostrada pelo algoritmo SMOTE Bordeline.

Figura 40 – Acurácia durante o treinamento e validação do ajuste fino realizado com a rede Xception e a base reamostrada pelo SMOTE Bordeline.

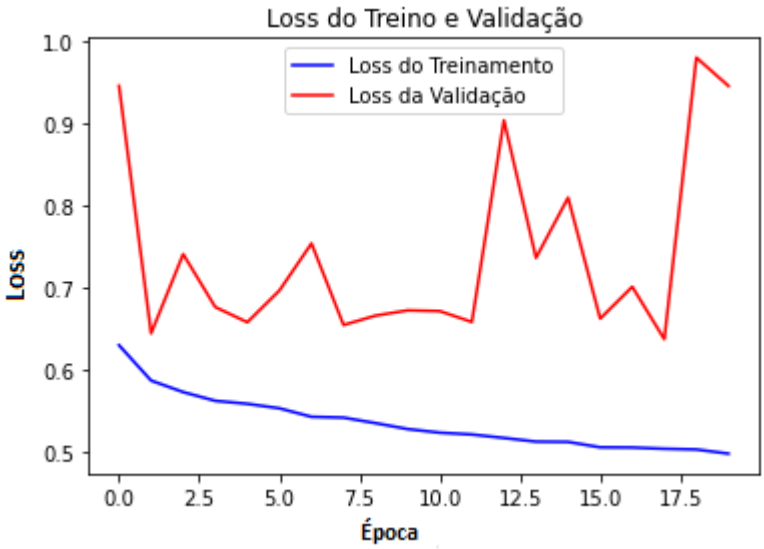


Fonte: O autor.

Como pode ser observado analisando o gráfico da Figura 40, durante o treinamento, o modelo foi se ajustando a cada iteração e melhorando a acurácia até a 18ª época. A acurácia variou entre 64% e 75%. Os dados de validação demostram uma acurácia que variou entre 45% e 66%.

A Figura 41 mostra o comportamento da loss durante o treinamento e validação com rede Xcpetion e a base reamostrada pelo algoritmo SMOTE Bordeline.

Figura 41 – Loss durante o treinamento e validação do ajuste fino realizado com a rede Xception e a base reamostrada pelo SMOTE Bordeline.

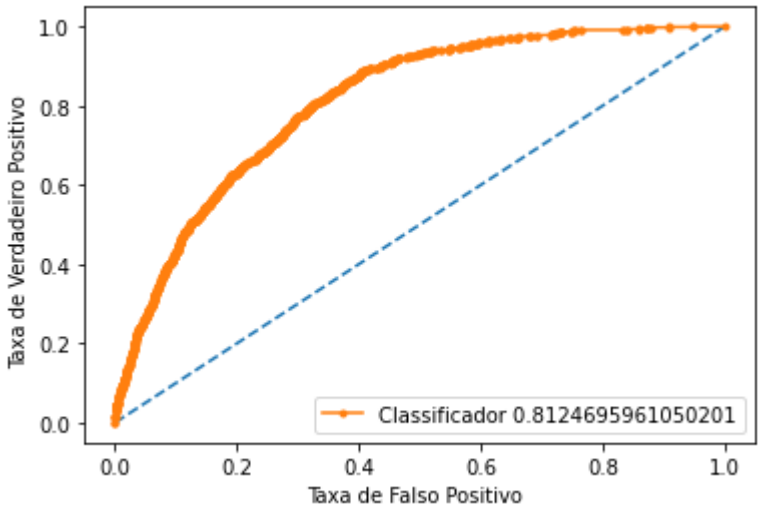


Fonte: O autor.

Pode-se observar na Figura 41 que a loss seguiu decaindo conforme o modelo se ajustava, iniciando em $0,65$ e chegando a aproximadamente $0,5$, onde não decaiu consideravelmente durante 5 épocas e o treinamento foi encerrado. A loss de validação variou entre $0,98$ e $0,65$.

A Figura 42 mostra curva ROC gerada através dos dados de teste, que apresentou uma área sob a curva de $81,2\%$ para verdadeiros positivos. Com a base de testes o modelo supracitado conseguiu uma acurácia de $58,33\%$.

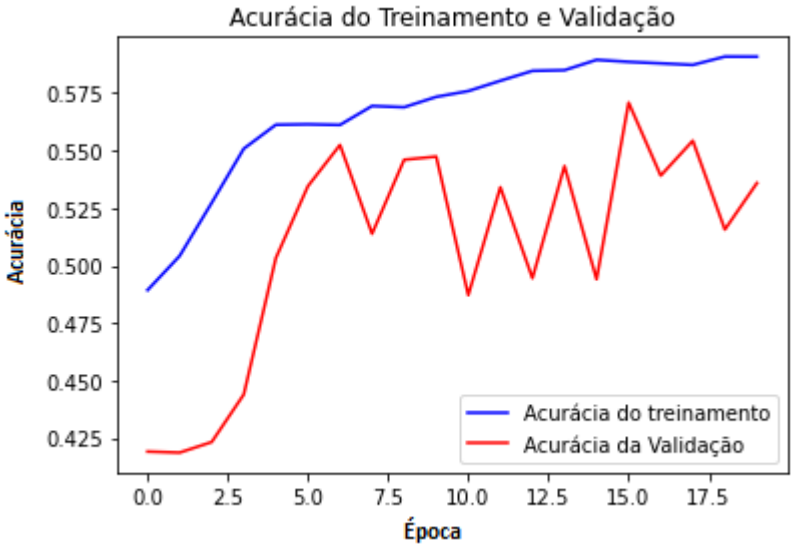
Figura 42 – Curva ROC gerada pelos dados de teste do ajuste fino realizado com a rede Xception e a base reamostrada pelo SMOTE Bordeline.



Fonte: O autor.

A Figura 43 representa ajuste fino utilizando a rede InceptionsResNetV2 com a base reamostrada pelo algoritmo SMOTE Adasyn durante o treinamento e validação.

Figura 43 – Acurácia durante o treinamento e validação do ajuste fino realizado com a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Adasyn.

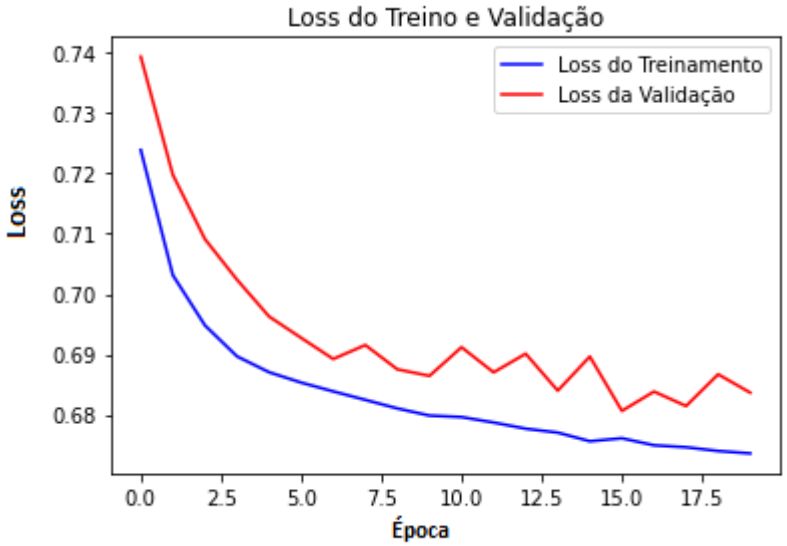


Fonte: O autor.

Esta configuração representada pela Figura 43 resultou em um modelo que foi se ajustando a cada iteração, melhorando a acurácia até a 18ª época, onde o treinamento foi finalizado, a acurácia variou entre 48% e 58%. Os dados de validação demostram uma acurácia instável variando entre 42% e 55%.

A Figura 44 mostra o comportamento da loss durante o treinamento e validação com rede InceptionsResNetV2 e a base reamostrada pelo algoritmo SMOTE Adasyn.

Figura 44 – Loss durante o treinamento e validação do ajuste fino realizado com a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Adasyn.

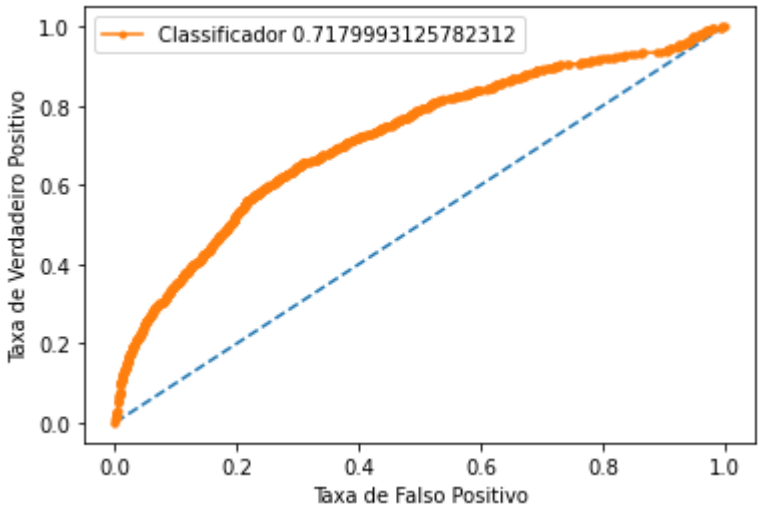


Fonte: O autor.

A Figura 44 demonstra que função loss decaiu conforme o modelo se ajustava, iniciando em $0,725$ e chegando a aproximadamente $0,67$, onde não decaiu consideravelmente durante 5 épocas e o treinamento foi encerrado. A perca de validação iniciou em $0,74$ e decaiu para cerca de $0,69$ durante o treinamento, indicando assim, uma dificuldade de aprendizado com a base utilizada.

A Figura 45 apresenta a curva ROC gerada através dos dados de teste apresentou uma área sob a curva de $71,7\%$ para verdadeiros positivos. Com a base de testes o modelo supracitado conseguiu uma acurácia de $63,07\%$.

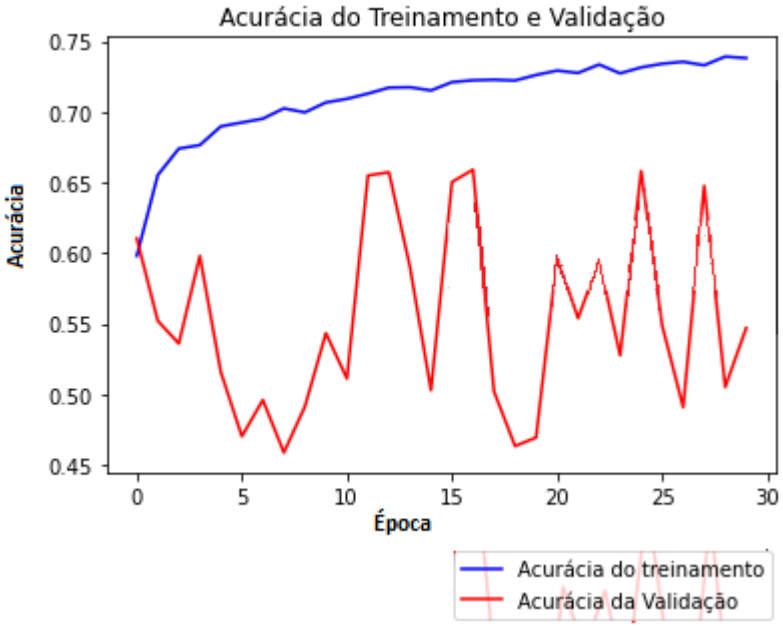
Figura 45 – Curva ROC gerada pelos dados de teste do ajuste fino realizado com a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Adasyn.



Fonte: O autor.

A Figura 46 demostra o resultado do ajuste fino utilizando a rede Xception com a base reamostrada pelo algoritmo SMOTE Adasyn durante o treinamento e validação.

Figura 46 – Acurácia durante o treinamento e validação do ajuste fino realizado com a rede Xception e a base reamostrada pelo SMOTE Adasyn.

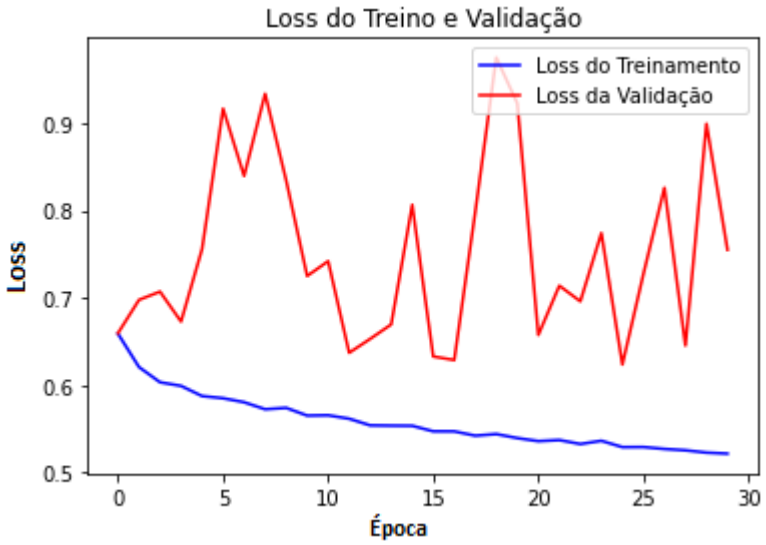


Fonte: O autor.

No treinamento (Figura 46), o modelo foi se ajustando a cada iteração, melhorando a acurácia até a 30ª época, onde o treinamento foi finalizado, a acurácia variou entre 60% e 74%. Os dados de validação demostram uma acurácia instável variando entre 46% e 65%.

A Figura 47 exibe a função de loss durante o treinamento e validação com rede Xception e a base reamostrada pelo algoritmo SMOTE Adasyn.

Figura 47 – Loss durante o treinamento e validação do ajuste fino realizado com a rede Xception e a base reamostrada pelo SMOTE Adasyn.

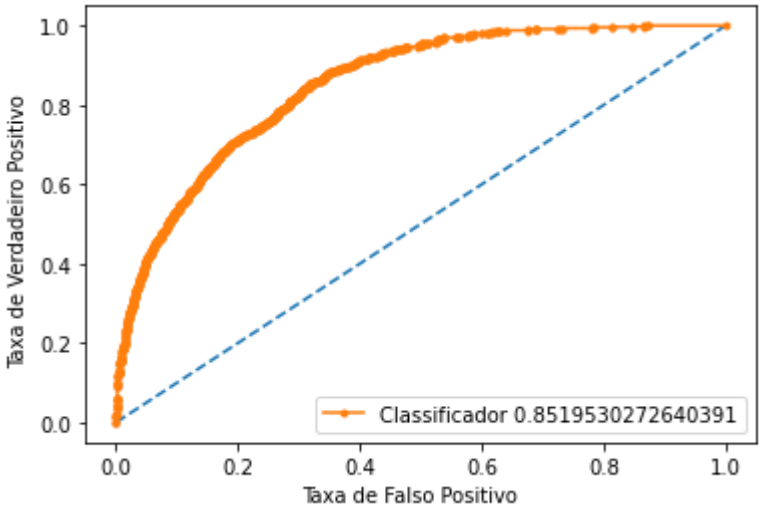


Fonte: O autor.

A loss representada pela Figura 47 decaiu conforme o modelo se ajustava, iniciando em $0,68$ e chegando a aproximadamente $0,55$, onde não decaiu consideravelmente durante 5 épocas e o treinamento foi encerrado. A loss de validação instável e maior que a de treinamento, indicou uma má generalização do modelo.

A figura Figura 48 demostra a curva ROC obtida utilizando a rede Xception e a base reamostrada pelo algoritmo SMOTE Adasyn.

Figura 48 – Curva ROC gerada pelos dados de teste do ajuste fino realizado com a rede Xception e a base reamostrada pelo SMOTE Adasyn.

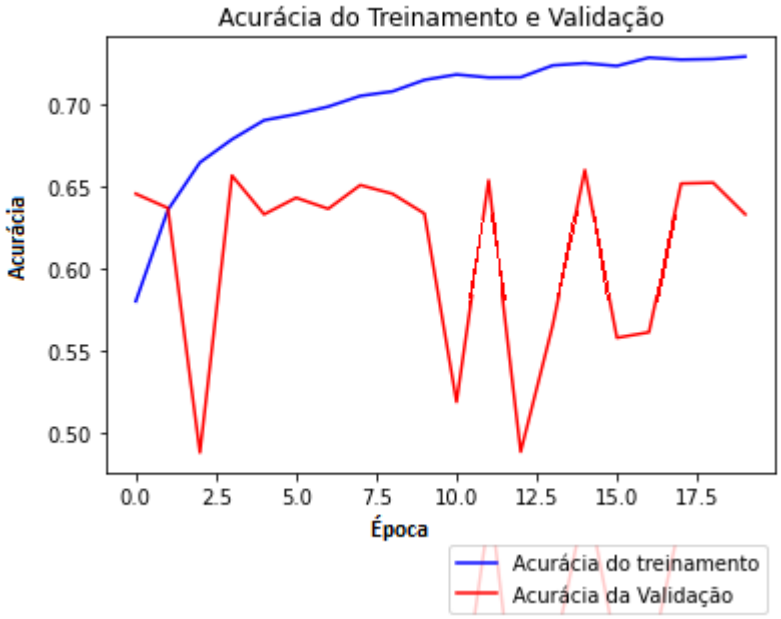


Fonte: O autor.

A Figura 48 apresenta curva ROC gerada através dos dados de teste que apresentou uma área sob a curva de $85,1\%$ para verdadeiros positivos. Com a base de testes o modelo supracitado conseguiu uma acurácia de $69,3\%$.

A Figura 49 ilustra o ajuste fino utilizando a rede InceptionsResNetV2 com a base reamostrada pelo algoritmo SMOTE Bordeline durante o treinamento e validação.

Figura 49 – Acurácia durante o treinamento e validação do ajuste fino realizado com a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Bordeline.

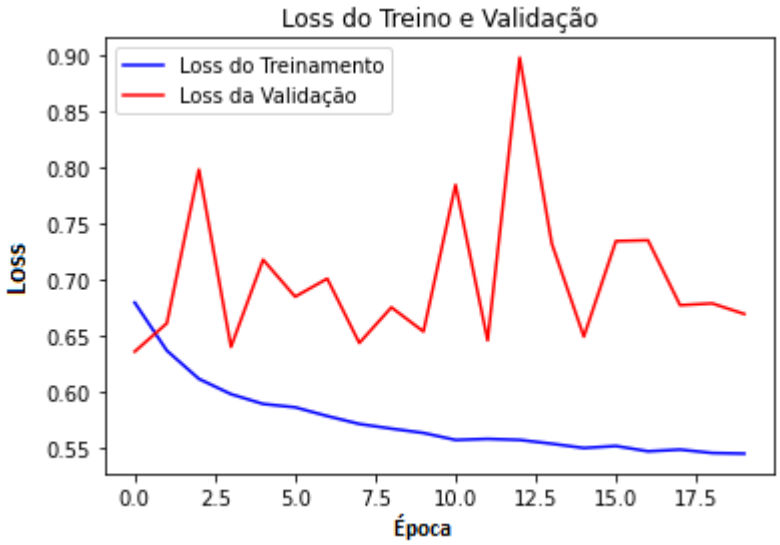


Fonte: O autor.

Pode-se observar na Figura 49 que durante treinamento, o modelo foi se ajustando a cada iteração, melhorando a acurácia até a 18ª época, onde o treinamento foi finalizado, a acurácia variou entre 58% e 74%. Os dados de validação demostram uma acurácia instável variando entre 45% e 65%.

A Figura 50 demonstra o comportamento da loss durante o treinamento e validação com rede InceptionsResNetV2 e a base reamostrada pelo algoritmo SMOTE Bordeline.

Figura 50 – Loss durante o treinamento e validação do ajuste fino realizado com a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Bordeline.

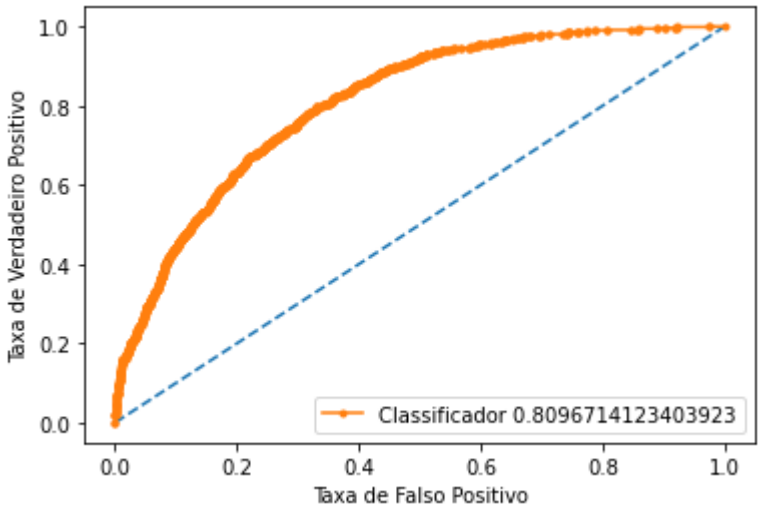


Fonte: O autor.

A Figura 50 ilustra a loss que decaiu conforme o modelo se ajustava, iniciando em $0,68$ e chegando a aproximadamente $0,55$, onde não decaiu consideravelmente durante 5 épocas e o treinamento foi encerrado. A loss de validação instável indicando dificuldade do modelo em classificar dados diferentes aos de treinamento.

A figura Figura 51 apresenta a curva ROC obtida utilizando a rede InceptionsResNetV2 e a base reamostrada pelo algoritmo SMOTE Bordeline.

Figura 51 – Curva ROC gerada pelos dados de teste do ajuste fino realizado com a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Bordeline.



Fonte: O autor.

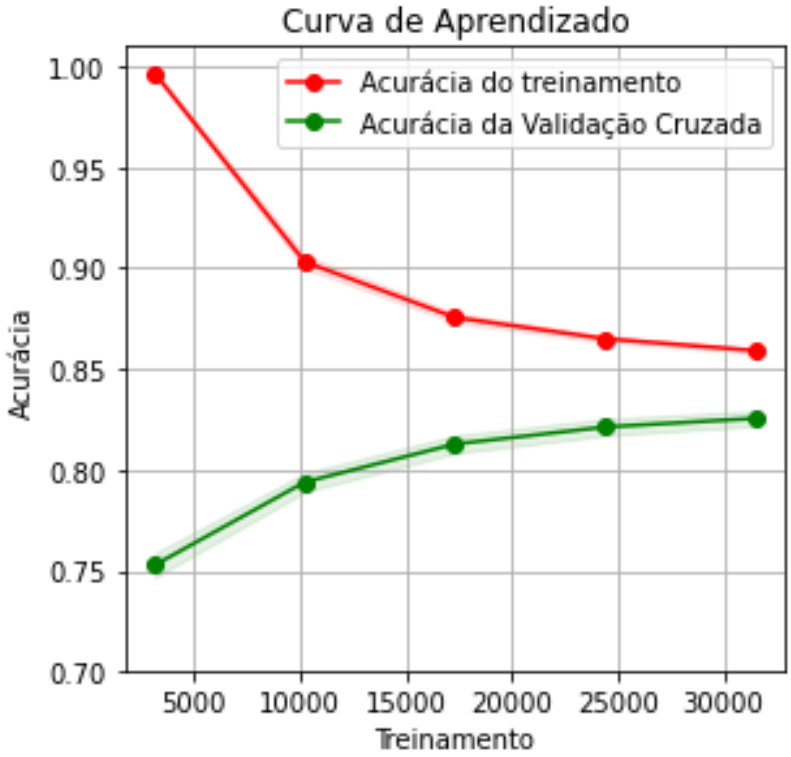
A Figura 51 exibe a curva ROC gerada através dos dados de teste apresentou uma área sob a curva de $80,9\%$ para verdadeiros positivos. Com a base de testes o modelo supracitado conseguiu uma acurácia de $75,94\%$.

6.2 Experimentos II (SVM)

Os resultados apresentados abaixo advém da transferência de aprendizado onde nenhuma camada dos modelos Xception e InceptionResNetV2 foram treinados, as camadas totalmente conectadas foram removidas, e todas camadas que permaneceram foram congeladas. As saídas das redes foram utilizadas como entrada para um classificador SVM Linear. Ambas as bases de treinamento resultantes dos algoritmos de balanceamento de base de dados foram utilizados. A loss utilizada foi a Hinge (a.k.a perda da dobradiça), e a validação cruzada foi realizada com partições com tamanho equivalente a 20% dos dados de treinamento e validação.

A Figura 52 mostra a evolução da transferência de aprendizado com a SVM substituindo a camada totalmente conectada para a base reamostrada com o SMOTE Adasyn e a rede Xcpetion.

Figura 52 – Curva de aprendizado gerada pela SVM que utilizou a base reamostrada pelo SMOTE Adasyn com as características extraídas da rede Xception.

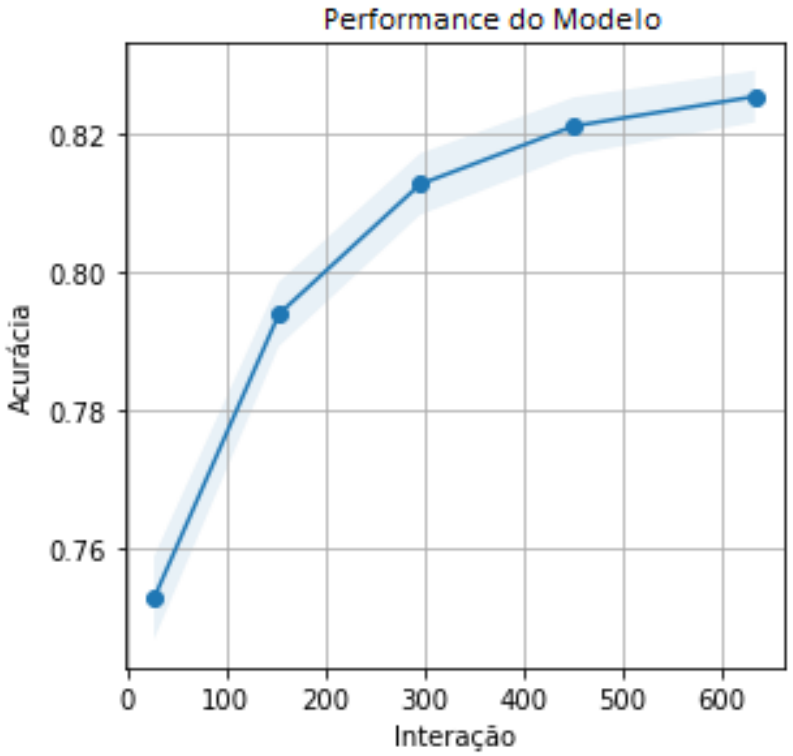


Fonte: O autor.

A Figura 52 demonstra que durante o treinamento, esta configuração iniciou-se com a acurácia máxima e decaiu conforme o volume de dados foi aumentando, terminando em cerca de 86%, a validação cruzada, por sua vez, iniciou com a acurácia de 75% e aumentou até aproximadamente 83%. Demonstrando que a complexidade do modelo aumentou conforme mais dados eram avaliados.

A Figura 53 ilustra a acurácia por iteração gerada pela SVM utilizando a base reamostrada com o SMOTE Adasyn e a rede Xcpetion.

Figura 53 – Curva de acurácia por iteração gerada pela SVM que utilizou a base reamostrada pelo SMOTE Adasyn com as características extraídas da rede Xception.

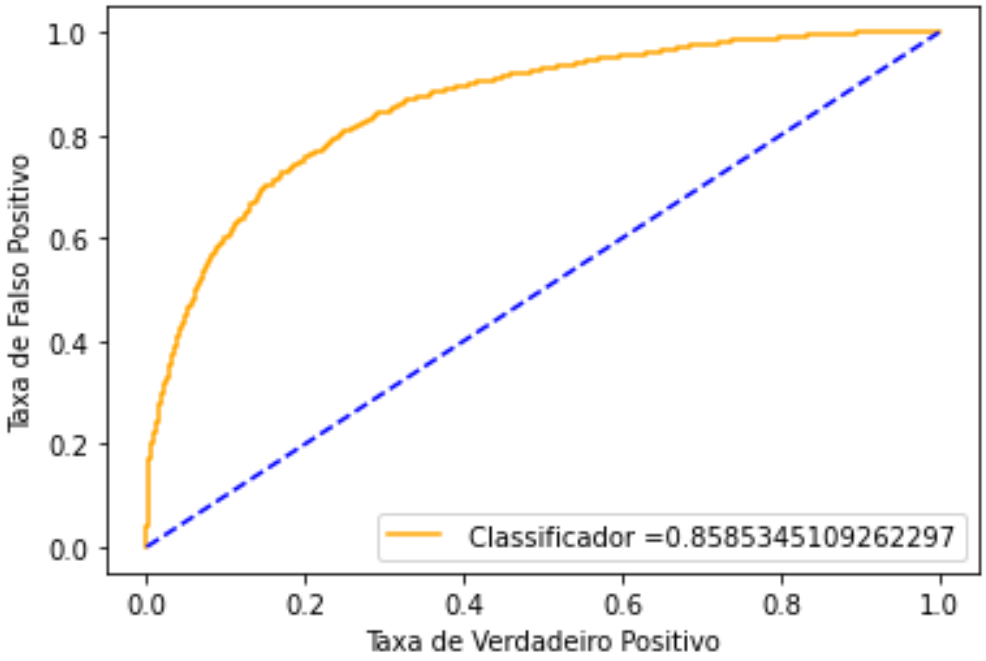


Fonte: O autor.

A acurácia do modelo nas primeiras iterações foi de 75% e atingiu 83% na 600ª iteração onde o aprendizado estabilizou e o treinamento foi encerrado, como ilustrado na Figura 53.

A Figura 54 apresenta a curva ROC gerada pela SVM substituindo a camada totalmente conectada para a base reamostrada com o SMOTE Adasyn e a rede Xcpetion.

Figura 54 – Curva ROC gerada pela SVM utilizando os dados de teste, que utilizou a rede Xception e a base reamostrada pelo SMOTE Adasyn no treinamento.

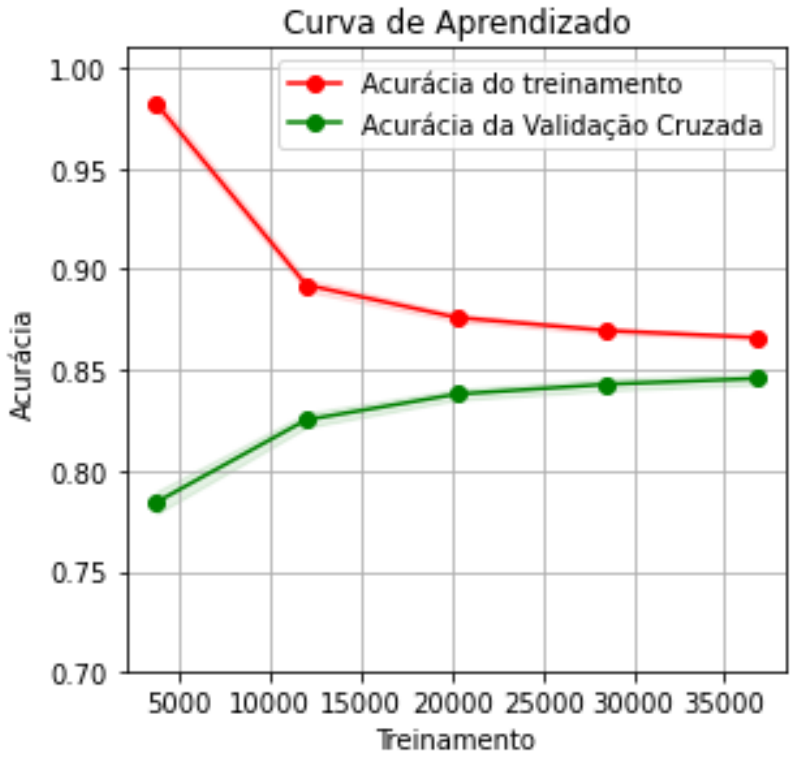


Fonte: O autor.

A curva ROC (Figura 54) apresentou uma área sob a curva de $85,8\%$ para verdadeiros positivos, para a SVM que utilizou a rede Xception e a base reamostrada pelo SMOTE Adasyn no treinamento. Ao realizar o teste do modelo com os dados da base de teste, foi aferida uma acurácia de $80,6\%$.

A Figura 55 mostra o comportamento da transferência de aprendizado com a SVM substituindo a camada totalmente conectada para a base reamostrada com o SMOTE Borderline e a rede InceptionResNetV2.

Figura 55 – Curva de aprendizado gerada pela SVM que utilizou a base reamostrada pelo SMOTE Borderline com as características extraídas da rede InceptionsResNetV2.

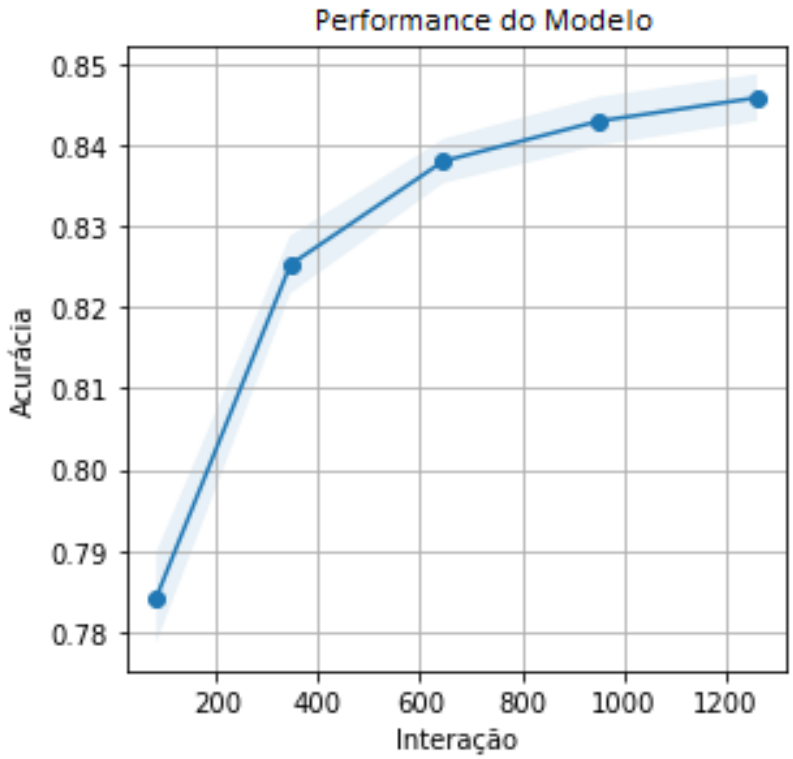


Fonte: O autor.

A Figura 55 ilustra o treinamento que iniciou-se com a acurácia de 98% e decaiu conforme o volume de dados aumentava, terminando em cerca de 86%, a validação cruzada, por sua vez, iniciou-se com a acurácia de $78,5\%$ e aumentou até aproximadamente $84,8\%$.

A Figura 56 ilustra a acurácia por iteração gerada pela SVM utilizando a base reamostrada com o SMOTE Borderline e a rede InceptionsResNetV2.

Figura 56 – Curva de acurácia por iteração gerada pela SVM que utilizou a base reamostrada pelo SMOTE Borderline com as características extraídas da rede InceptionsResNetV2.

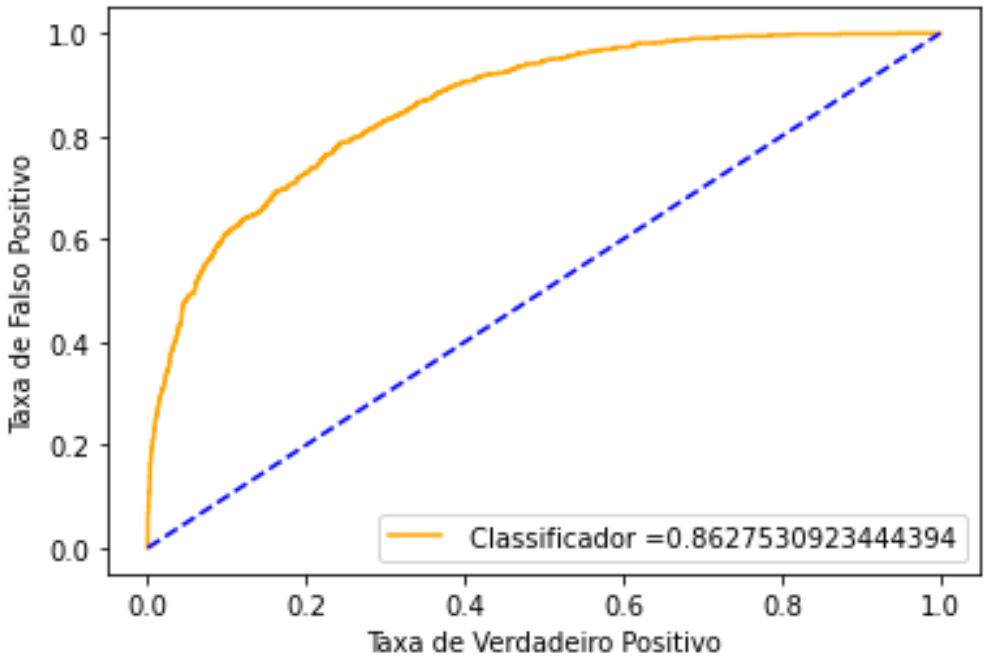


Fonte: O autor.

A acurácia do modelo nas primeiras iterações foi de $78,5\%$ e atingiu $84,8\%$, na 1200^{a} iteração onde o aprendizado estabilizou e o treinamento foi encerrado, como pode ser verificado na Figura 56.

A Figura 57 apresenta a curva ROC gerada pela SVM substituindo a camada totalmente conectada para a base reamostrada com o SMOTE Borderline e a rede Inceptions-ResNetV2.

Figura 57 – Curva ROC gerada pela SVM utilizando os dados de teste, que utilizou a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Borderline no treinamento.

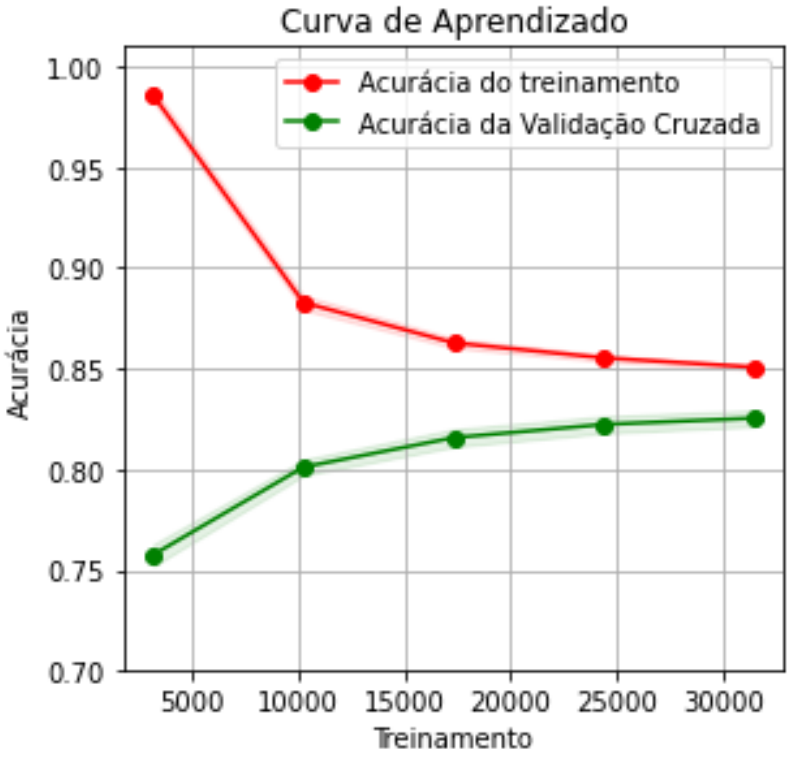


Fonte: O autor.

Pode-se observar na Figura 57 que a curva ROC apresentou uma área sob a curva de $86,2\%$ para verdadeiros positivos, para a SVM que utilizou a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Borderline no treinamento. Ao realizar o teste do modelo com os dados da base de teste, foi aferida uma acurácia de $80,6\%$.

A Figura 58 mostra a evolução da transferência de aprendizado com a SVM substituindo a camada totalmente conectada para a base reamostrada com o SMOTE Adasyn e a rede InceptionsResNetV2.

Figura 58 – Curva de aprendizado gerada pela SVM que utilizou a base reamostrada pelo SMOTE Adasyn com as características extraídas da rede InceptionsResNetV2.

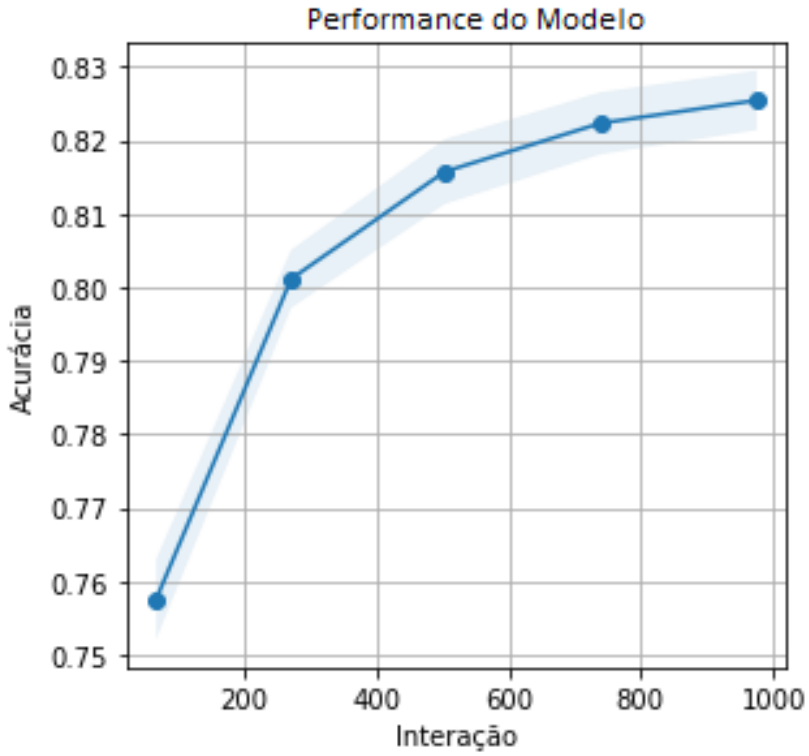


Fonte: O autor.

O treinamento da SVM, utilizando as características extraídas da InceptionsResNetV2 (Figura 58), iniciou-se com a acurácia 98% e decaiu conforme o volume de dados foi aumentando, terminando em cerca de 85%, a validação cruzada, no entanto, iniciou-se com a acurácia de $75,9\%$ e aumentou até aproximadamente $82,5\%$.

A Figura 59 ilustra a acurácia por iteração gerada pela SVM utilizando a base reamostrada com o SMOTE Adasyn e a rede InceptionsResNetV2.

Figura 59 – Curva de acurácia por iteração gerada pela SVM que utilizou a base reamostrada pelo SMOTE Adasyn com as características extraídas da rede InceptionsResNetV2.

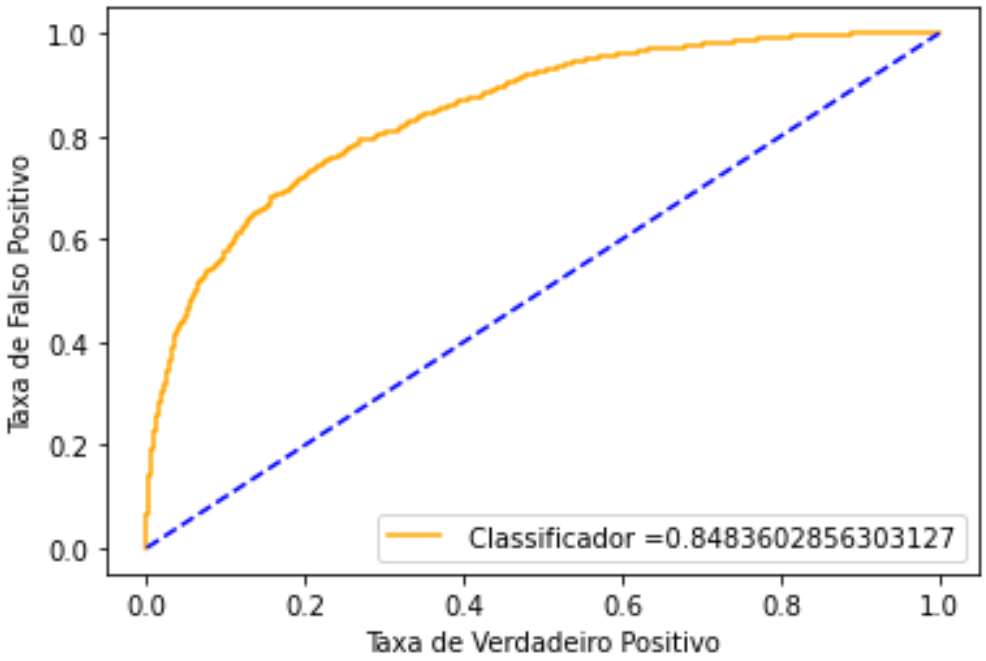


Fonte: O autor.

A acurácia do modelo nas primeiras iterações foi de $75,9\%$ e atingiu $82,5\%$, na 800^{a} iteração onde o aprendizado estabilizou e o treinamento foi encerrado, como apresentado na Figura 59.

A Figura 60 apresenta a curva ROC gerada pela SVM substituindo a camada totalmente conectada para a base reamostrada com o SMOTE Adasyn e a rede Inceptions-ResNetV2.

Figura 60 – Curva ROC gerada pela SVM utilizando os dados de teste, que utilizou a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Adasyn no treinamento

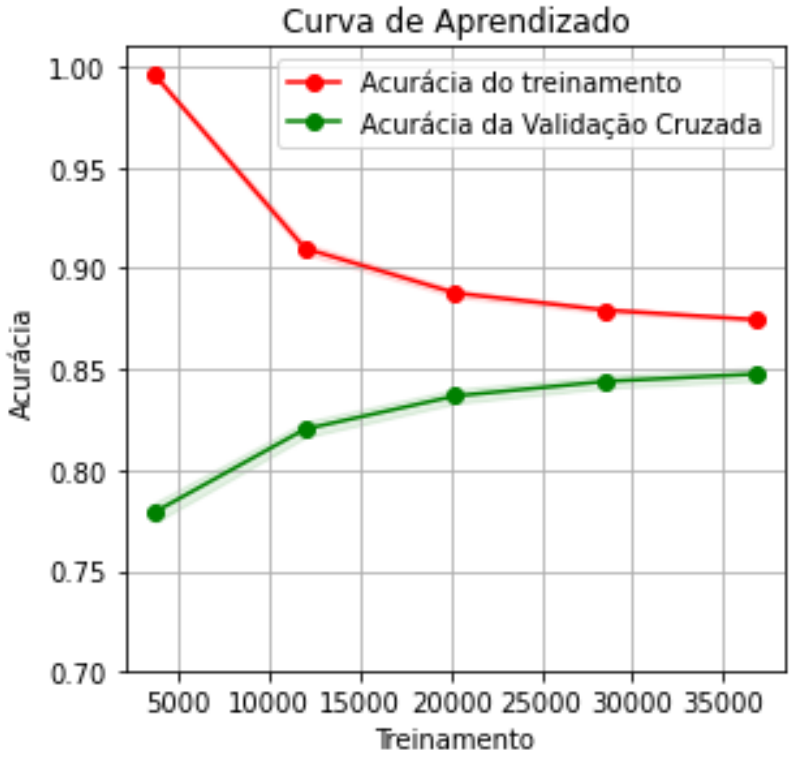


Fonte: O autor.

A curva ROC, ilustrada na Figura 60, apresentou uma área sob a curva de $84,8\%$ para verdadeiros positivos, para a SVM que utilizou a rede InceptionsResNetV2 e a base reamostrada pelo SMOTE Adasyn no treinamento. Ao realizar o teste do modelo com os dados da base de teste, foi aferida uma acurácia de $81,14\%$.

A Figura 61 mostra a evolução da transferência de aprendizado com a SVM substituindo a camada totalmente conectada para a base reamostrada com o SMOTE Borderline e a rede Xcpetion.

Figura 61 – Curva de aprendizado gerada pela SVM que utilizou a base reamostrada pelo SMOTE Borderline com as características extraídas da rede Xception.

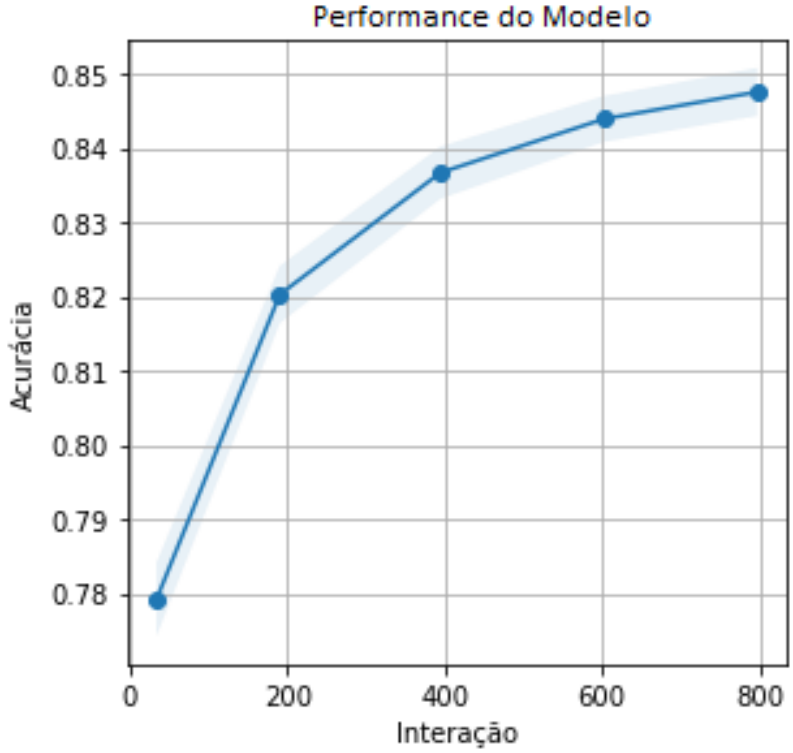


Fonte: O autor.

Pode-se observar na Figura 61 que durante o treinamento desta configuração o algoritmo iniciou-se com a acurácia máxima e decaiu conforme o volume de dados aumentava, terminando em cerca de 87%, a validação cruzada, por sua vez, iniciou-se com a acurácia de 78% e aumentou até aproximadamente 84,9%.

A Figura 62 ilustra a acurácia por iteração gerada pela SVM utilizando a base reamostrada com o SMOTE Borderline e a rede Xcpetion.

Figura 62 – Curva de acurácia por iteração gerada pela SVM que utilizou a base reamostrada pelo SMOTE Borderline com as características extraídas da rede Xception.

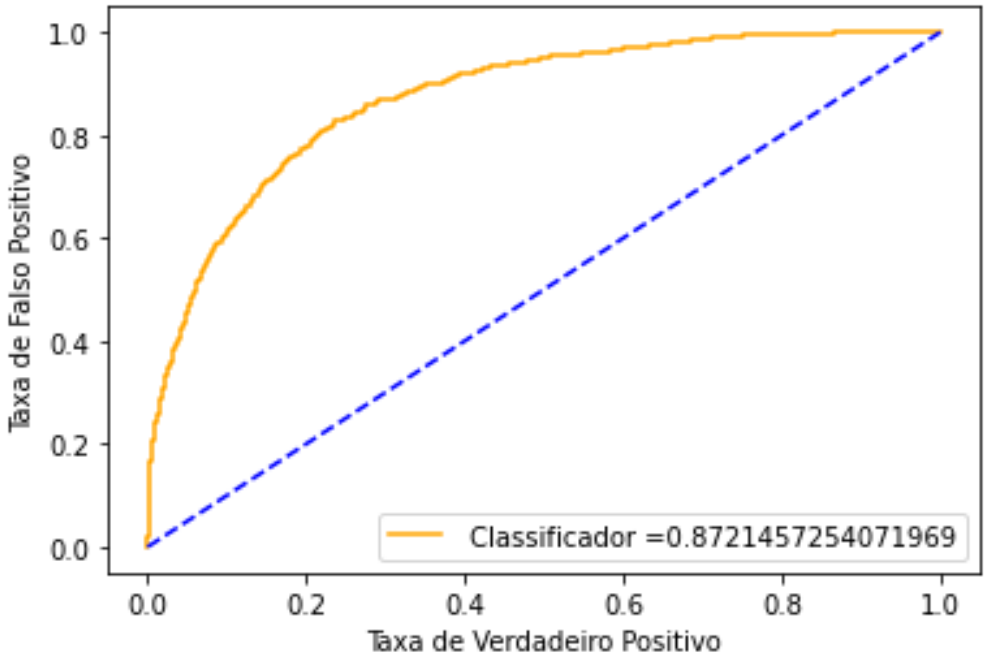


Fonte: O autor.

A acurácia do modelo (Figura 62) nas primeiras iterações foi de 78% e atingiu 84, 9%, na 800ª iteração onde o aprendizado estabilizou e o treinamento foi encerrado.

A Figura 63 apresenta a curva ROC gerada pela SVM substituindo a camada totalmente conectada para a base reamostrada com o SMOTE Borderline e a rede Xcpetion.

Figura 63 – Curva ROC gerada pela SVM utilizando os dados de teste, que utilizou a rede Xception e a base reamostrada pelo SMOTE Borderline no treinamento



Fonte: O autor.

A curva ROC exibida na Figura 63 apresentou uma área sob a curva de 87, 2% para verdadeiros positivos, para a SVM que utilizou a rede Xception e a base reamostrada pelo SMOTE Borderline no treinamento. Ao realizar o teste do modelo com os dados da base de teste, foi aferida uma acurácia de 81, 15%.

6.3 Discussão

Nesta seção, discutimos os resultados dos 2 (dois) experimentos anteriores. Para tanto, avaliamos cada experimento e comparamos os resultados obtidos com a literatura.

A Tabela 2 mostra de comparação dos resultados dos dois experimentos realizados;

Tabela 2 – Resumo geral dos experimentos realizados neste estudo.

Rede	Algoritmo de Reamostragem	Classificador	Acurácia de Treinamento	Acurácia de Teste	AUC ROC
Xception	Borderline	Camada totalmente conectada	75%	58,33%	81,2%
InceptionsResNetV2	Adasyn	Camada totalmente conectada	58%	63,07%	71,7%
Xception	Adasyn	Camada totalmente conectada	74%	69,3%	85,1%
InceptionsResNetV2	Borderline	Camada totalmente conectada	74%	75,94%	80,9%
Xception	Adasyn	SVM	86%	80,6%	85,8%
InceptionsResNetV2	Borderline	SVM	86%	80,6%	86,2%
InceptionsResNetV2	**Adasyn**	**SVM**	**85%**	**81,14%**	**84,8%**
Xception	**Borderline**	**SVM**	**87%**	**81,15%**	**87,2%**

Fonte: O autor.

Os resultados da Tabela 2 foram ordenados por ordem de acurácia de teste. Estes resultados mostram que a transferência de aprendizagem foi inferior ao SVM.

Note que as bases utilizadas no ajuste fino são dissemelhantes da base original em que os modelos utilizados foram treinados, assim, as características aprendidas durante a maior parte do treinamento original das redes, não descrevem bem os dados utilizados em nossos experimentos. Note também que realizar um treinamento de ponta a ponta das redes como em (ESTEVA et al., 2017) não seria viável, pois segundo (STANFORD, 2015) para este treinamento seria necessário um volume grande de dados disponíveis. Em (ESTEVA et al., 2017), por exemplo, os pesquisadores tiveram acesso a 3 bases de imagens de lesões de pele, que totalizavam 129.450 imagens, formando um volume aproximadamente 3 vezes maior que as bases utilizadas nesta pesquisa.

Os experimentos com SVM (veja Seção 6.2) foram consistentes e a acurácia relativamente alta. As variações que utilizaram o SVM resultaram em acurácias bastante próximas, com destaque para a configuração que utilizou a rede Xception, com a base de dados reamostrada pelo algoritmo SMOTE Borderline, que apresentou as melhores acurácias durante o treinamento e com a base de teste, também apresentou a melhor curva ROC e AUC. Isso acontece porque o classificador SVM funciona melhor que a transferência de aprendizado quando a base de dados é pequena (STANFORD, 2015).

A comparação dos classificadores com a literatura foi realizada tomando como *baseline* o trabalho em (ESTEVA et al., 2017). A Tabela 3 mostra as acurácias dos modelos desenvolvidos nesse estudo e (ESTEVA et al., 2017). Essa avaliação foi realizada calculando a média das inferências individuais de cada classe pelo modelo, onde os dois melhores modelos deste trabalho (veja Tabela 2) foram comparados com os dois melhores modelos apresentados na literatura de referência.

O segundo melhor modelo atingiu individualmente para as imagens da base de teste para a classe benigna a acurácia de 84,19% e para as malignas 68,17%, atingindo assim a

média de $76,18\%$. O melhor modelo atingiu individualmente para as imagens da base de teste para a classe benigna a acurácia de $82,76\%$ e para as malignas $74,32\%$, atingindo assim a média de $78,54\%$.

Os modelos supracitados, são denominados, respectivamente, InceptionResNetV2-Adasyn-SVM e Xception-Borderline-SVM apenas para facilitar a compreensão da comparação dos melhores resultados deste estudo com os obtidos em (ESTEVA et al., 2017).

Tabela 3 – Comparação dos resultados em Nature e o melhor modelo gerado neste estudo.

Modelo	Acurácia média das inferências individuais
CNN - Nature	$69,4\%$
CNN PA - Nature	$72,1\%$
InceptionResNetV2-Adasyn-SVM	$76,18\%$
Xception-Borderline-SVM	$78,54\%$

Fonte: (ESTEVA et al., 2017) e o autor.

Vale ressaltar que os modelos de (ESTEVA et al., 2017), tinham como objetivo classificar as imagens em lesões benignas, malignas e não neoplásicas, enquanto os modelos deste trabalho classificar as imagens em benignas e malignas.

Apesar da quantidade de classes classificadas, bases, métodos de treinamento e classificadores diferentes, é possível afirmar que a hipótese de que redes neurais artificiais são artifícios promissores para a classificação de imagens de lesões de pele em candidatos a câncer.

Capítulo 7

Conclusões e Trabalhos Futuros

Este trabalho propôs uma abordagem que utiliza redes neurais artificiais para realizar a predição de lesões de pele em possíveis candidatos a cânceres de pele. Em particular, definimos como referência a pesquisa realizada em (ESTEVA et al., 2017).

Os resultados obtidos foram satisfatórios e os objetivos foram alcançados, utilizando uma combinação da transferência de aprendizado das redes consideradas estado da arte, técnicas de reamostragem de dados e classificação através de funções SVM.

Como resultado este estudo reforça que as redes neurais artificiais são recursos promissores para acelerar a identificação de câncer de pele, uma vez que, o diagnóstico precoce é uma variável primordial que aumenta a taxa de sobrevivência deste tipo de enfermidade.

Também foi possível concluir que, atualmente, o principal desafio para a evolução deste tipo de modelo, é a quantidade limitada de dados validadas por biópsia, principalmente, para os dados de lesões malignas, restringindo assim, um treinamento mais efetivo e generalista.

Para possíveis trabalhos futuros, é preciso considerar reunir mais imagens de lesões certificadas para o treinamento de ponta a ponta das redes neurais convolucionais consideradas estado da arte, também como, utilizar métodos de aumento de dados variados, realizar a classificação por enfermidades de pele mais específicas e realizar a disponibilização do modelo para aplicação na vida real, podendo assim, coletar a acurácia deste tipo de modelo na prática e ao mesmo tempo popularizar a utilização deste tipo de ferramenta.

Referências

AL-MEJIBLI, I. S.; ALWAN, J. K.; ABD, D. H. The effect of gamma value on support vector machine performance with different kernels. **International Journal of Electrical and Computer Engineering (IJECE)**, v. 10, p. 5497, 2020.

ALEMI, A. **Improving Inception and Image Classification in TensorFlow**. [S.l.], 2016. Disponível em: <https://ai.googleblog.com/2016/08/improving-inception-and-image.html>. Acesso em: 17 de abr. de 2021.

AMAMI, R.; AYED, D. B.; ELLOUZE, N. Practical selection of svm supervised parameters with different feature representations for vowel recognition. v. 7, 2015.

AMIDI, A.; AMIDI, S. **Convolutional Neural Networks cheatsheet**. [S.l.], 2018. Disponível em: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>. Acesso em: 04 de fev. de 2021.

ATLAS. **Atlas On-line de Mortalidade**. Brasília, DF, 2019. Disponível em: <https://mortalidade.inca.gov.br/MortalidadeWeb/>. Acesso em: 07 de dez. de 2020.

ATLAS. **Atlas On-line de Mortalidade**. Brasília, DF, 2020. Disponível em: <https://www.inca.gov.br/MortalidadeWeb/pages/Modelo10/consultar.xhtml>. Acesso em: 07 de dez. de 2020.

BACKES, A. R.; JUNIOR, J. J. de M. S. **Introdução à Visão Computacional Usando MATLAB**. Rio de Janeiro, RJ: ALTA BOOKS, 2016.

BRASIL. **Câncer de pele: o que é, causas, sintomas, tratamento e prevenção**. Brasília, DF, 2019. Disponível em: <https://antigo.saude.gov.br/saude-de-a-z/cancer-de-pele>. Acesso em: 05 de dez. de 2020.

BROWNNLEE, J. **Develop Deep Learning Models on Theano and TensorFlow**. 1.8. ed. Melbourne, Australia: Machine Learning Mastery, 2017.

BUSSAB, W. D. O.; MORETTIN, P. A. **Estatística Básica**. 6. ed. São Paulo, SP: Saraiva, 2010.

CHAWLA, N. V. et al. Smote: Synthetic minority over-sampling technique. **J. Artif. Intell. Res. (JAIR)**, v. 16, p. 321–357, 2002.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 1800–1807.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, 1995.

DEVELOPERS, G. **Machine Learning Crash Course Courses Practica Guides Glossary - Classification: Accuracy**. [S.l.], 2018. Disponível em: <https://developers.google.com/machine-learning/crash-course>. Acesso em: 24 de abr. de 2021.

DEVELOPERS, G. **Machine Learning Crash Course Courses Practica Guides Glossary - Classification: ROC Curve and AUC**. [S.l.], 2018. Disponível em: <https://developers.google.com/machine-learning/crash-course>. Acesso em: 26 de abr. de 2021.

DEVELOPERS, G. **Machine Learning Crash Course Courses Practica Guides Glossary - Descending into ML: Training and Loss**. [S.l.], 2018. Disponível em: <https://developers.google.com/machine-learning/crash-course>. Acesso em: 24 de abr. de 2021.

EINSTEIN, H. A. **Google e Einstein lançam parceria para resultados de busca sobre saúde**. [S.l.], 2016. Disponível em: <https://www.einstein.br/noticias/noticia/google-einstein-lancam-parceria-para-resultados-busca-sobre-saude>. Acesso em: 20 de nov. de 2020.

ESTEVA, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. **Nature**, v. 542, n. 7639, p. 115–118, 2017.

FAWCETT, T. Introduction to roc analysis. **Pattern Recognition Letters**, v. 27, p. 861–874, 2006.

FOTOS, G. **Armazenamento gratuito e organização automática para todas as suas memórias.** [S.l.], 2021. Disponível em: <https://www.google.com/intl/pt-BR/photos/about/>. Acesso em: 01 de abr. de 2021.

HAENLEIN, M.; KAPLAN, A. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. **California Management Review**, v. 61, n. 4, p. 5–14, 2019.

HAN, H.; WANG, W.-Y.; MAO, B.-H. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In: **Advances in Intelligent Computing**. [S.l.]: Springer Berlin Heidelberg, 2005. p. 878–887.

HE, H. et al. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: . [S.l.: s.n.], 2008. p. 1322 – 1328.

INCA. **Câncer de pele melanoma**. Brasília, DF, 2021. Disponível em: <https://www.inca.gov.br/tipos-de-cancer/cancer-de-pele-melanoma>. Acesso em: 05 de abr. de 2021.

INCA. **Câncer de pele não melanoma**. Brasília, DF, 2021. Disponível em: <https://www.inca.gov.br/tipos-de-cancer/cancer-de-pele-nao-melanoma>. Acesso em: 05 de abr. de 2021.

ISIC. **About ISIC**. [S.l.], 2018. Disponível em: <https://www.isic-archive.com/#!/topWithHeader/tightContentTop/about/isicArchive>. Acesso em: 15 de out. de 2020.

LENS, G. **Pesquise o que você vê**. [S.l.], 2021. Disponível em: <https://lens.google.com/intl/pt-BR_br/>. Acesso em: 01 de abr. de 2021.

NWANKPA, C. E. et al. Activation functions: Comparison of trends in practice and research for deep learning. **2nd International Conference on Computational Sciences and Technologies**, p. 124 – 133, 2020.

OGATA, K. **Engenharia de Controle Moderno**. 5. ed. Rio de Janeiro, RJ: Pearson / Prentice Hall, 2010.

PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. Deep face recognition. In: **British Machine Vision Conference**. [S.l.: s.n.], 2015.

PELTARION. **Categorical crossentropy**. [S.l.], 2020. Disponível em: <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy>. Acesso em: 24 de abr. de 2021.

PENG, W. et al. Ida-3d: Instance-depth-aware 3d object detection from stereo vision for autonomous driving. In: **IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

PINHEIRO, J. I. D. et al. **Estatística Básica a arte de trabalhar com dados**. Rio de Janeiro, RJ: Elsevier, 2009.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. In: _____. **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 532–538.

RENNIE, J. D. M.; SREBRO, N. Loss functions for preference levels: Regression with discrete ordered labels. **Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling**, 2005.

ROTEMBERG, V. et al. A patient-centric dataset of images and metadata for identifying melanomas using clinical context. **Scientific Data**, v. 8, n. 1, p. 34, 2021.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. Rio de Janeiro, RJ: Elsevier, 2013.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: **International Conference on Learning Representations**. [S.l.: s.n.], 2015.

SINHA, P.; RUSSELL, R. A perceptually based comparison of image similarity metrics. **Perception**, v. 40, p. 1269–81, 2011.

SOCIETY, A. C. **Survival Rates for Melanoma Skin Cancer**. [S.l.], 2019. Disponível em: <https://www.cancer.org/cancer/melanoma-skin-cancer/detection-diagnosis-staging/survival-rates-for-melanoma-skin-cancer-by-stage.html>. Acesso em: 05 de dez. de 2020.

SOCIETY, C. C. **Survival statistics for non-melanoma skin cancer**. [S.l.], 2017. Disponível em: <https://www.cancer.ca/en/cancer-information/cancer-type/skin-non-melanoma/prognosis-and-survival/survival-statistics/>. Acesso em: 05 de dez. de 2020.

SOMASUNDARAM, A.; REDDY, U. S. Data imbalance: Effects and solutions for classification of large and highly imbalanced data. In: . [S.l.: s.n.], 2016.

STANFORD. **CS231n Convolutional Neural Networks for Visual Recognition**. [S.l.], 2015. Disponível em: <https://cs231n.github.io/convolutional-networks/>. Acesso em: 02 de fev. de 2021.

STANFORD. **CS231n Convolutional Neural Networks for Visual Recognition**. [S.l.], 2015. Disponível em: <https://cs231n.github.io/transfer-learning/>. Acesso em: 18 de mar. de 2021.

SZEGEDY, C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning. **AAAI Conference on Artificial Intelligence**, 2016.

SZEGEDY, C. et al. **Rethinking the Inception Architecture for Computer Vision**. 2015.

TSCHANDL, P. et al. Melanomas vs. nevi in high-risk patients under long-term monitoring with digital dermatoscopy: do melanomas and nevi already differ at baseline **J Eur Acad Dermatol Venereol**, p. 972–977, 2017.

YADAV, S. S.; JADHAV, S. M. Deep convolutional neural network based medical image classification for disease diagnosis. **Journal of Big Data**, v. 6, n. 1, p. 113, 2019.

ZHANG, R. et al. The unreasonable effectiveness of deep features as a perceptual metric. In: . [S.l.: s.n.], 2018. p. 586–595.