Classificação de Raças de Cachorro

https://github.com/isabelatelles/mc906_collab/p4
https://www.youtube.com/watch?v=YnLmpSr-nBArel=0

GIOVANNA VENDRAMINI
RA 173304
E-mail: g173304@dac.unicamp.br

ISABELA TELLES FURTADO DOSWALDO
RA 170012
E-mail: i170012@dac.unicamp.br

NATHÁLIA HARUMI KUROMIYA
RA 175188
E-mail: n175188@dac.unicamp.br

***Resumo* – O trabalho descreve o problema de predizer a raça de um cachorro através de uma imagem. Para solucionar tal problema, foram utilizadas redes neurais convolucionais pré-treinadas, e adicionadas camadas Flatten ou GlobalAveragePooling2D, Dense e Dropout, bem como ReLU e Softmax para ativação, através da biblioteca Keras. Foi feita a análise de resultados com base no valor de acurácia como métrica de avaliação e a comparação ocorreu variando o dataset, a rede neural pré-treinada, as camadas do topo, e o otimizador. A melhor solução obtida foi utilizando o segundo dataset, com a rede InceptionResNetV2 adicionada à camada Flatten, e otimizador Adam, que forneceu uma acurácia de validação de 93,50%. Para o primeiro dataset, obteve-se 83,02% de acurácia de validação ao reduzi-lo para 35 classes, utilizar também a rede InceptionResNetV2, mas adicionada à camada GlobalAveragePooling2D, e o otimizador Adam.**

***Palavras-chave* – Redes Neurais, IA, CNN, raça, cachorro**

I. INTRODUÇÃO

O intuito deste trabalho é construir um modelo capaz de classificar raças de cachorros a partir de uma imagem dada. Para a confecção do trabalho, foram estudados os conceitos de aprendizado de máquina e redes neurais, apresentados nos slides [1], [2], [3] e [4] da disciplina MC906.

Uma rede neural artificial é composta por neurônios que, ao receberem entradas e darem a elas diferentes pesos, é capaz de realizar o aprendizado de máquina e reconhecer padrões [5]. A motivação para esse estudo está no fato de redes neurais, na maioria das vezes, apresentarem um bom desempenho na resolução de problemas, rápida implementação e adaptabilidade. Em especial, redes neurais convolucionais (CNNs) tem sido muito utilizadas em aplicações como detecção e classificação de objetos. Dado esse fator, foi feita a escolha do problema de classificação de raças de cachorros, que é por si só um problema desafiador até mesmo para humanos, dada a diversidade de raças existentes e a semelhança de características entre algumas delas.

Esse é um tema recorrente em pesquisas e foi também tema de uma competição proposta pelo Kaggle[6]. Os estudos anteriores relacionados serviram de inspiração para implementação da solução apresentada neste trabalho, bem como para comparação de resultados e abordagens do problema.

Um dos trabalhos realizados nessa área é o ”Dog Breed Identification”[7], que tem como intuito realizar a identificação da raça combinando aprendizado de máquina e visão computacional, uma vez que determina os pontos-chave da face do animal através de uma rede neural e depois utiliza o descritor SIFT para extrair suas features. Foram encontradas outras duas abordagens muito semelhantes à descrita neste trabalho, que são as referentes a ”Dog Breed Identification”[8] e ”Modified Deep Neural Networks for Dog Breeds Identication”[9], sendo ambas solucionadas através do uso de redes neurais convolucionais pré-treinadas e da adição de camadas do topo.

Este trabalho está separado nas seguintes seções: I. Introdução, II. Trabalho Proposto, III. Metodologia, IV. Resultados e Discussão, V. Conclusões e Referências.

II. TRABALHO PROPOSTO [10]

O trabalho em questão propõe o uso de CNNs, redes neurais convolucionais, pré-treinadas para realizar a predição da raça ao fornecer como entrada a imagem de um cachorro.

Foram, então, realizadas análises e comparações entre diferentes datasets, CNNs e otimizadores, buscando determinar a solução de maior acurácia.

III. METODOLOGIA

O pré-processamento das imagens, a modelagem e o treinamento do modelo foram feitos utilizando Python 3[11], além das bibliotecas Keras[12], Scikit-learn[13], Numpy[14] e Pandas[15]. A plataforma utilizada para o treinamento foi o

Google Colab, que disponibiliza uma GPU com 12.72 GB de memória RAM e 358.27 GB de disco.

A. Datasets

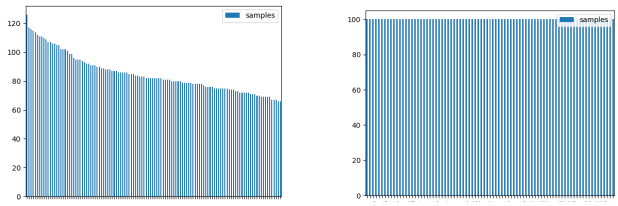
1) Dataset 1[6]: Esse *dataset* conta com 10222 exemplares de imagens para treino anotados e 10357 imagens para teste sem anotações. Ele é baseado no Stanford Dogs dataset.

Trata-se de um *dataset* desbalanceado, onde há classes com 66 examplares até classes com mais de 120 exemplares. Por essa razão, e, principalmente, porque pelo menos metade das classes possuem menos de 80 exemplares, como pode ser visto na Figura 1, foi decidido que neste trabalho seria utilizado apenas as 35 classes com mais exemplares. Assim, essas foram balanceadas com apenas 90 examplares cada.

2) Dataset 2[16]: Trata-se de um *dataset* com 8300 imagens de cachorros para treino, onde há 83 classes com 100 exemplares cada, portanto, já é previamente balanceado. Para validação, ele conta com 6059 imagens, onde cada classe possui 73 exemplares. Para teste, há 5420 imagens.

Nesse dataset, todas as imagens possuem pelo menos 1 cachorro, porém podem possuir pessoas, mais de um cachorro ou qualquer outro elemento adicional na imagem.

Ele foi fornecido pelo professor *Anderson de Rezende Rocha* para efeito de estudos.



(a) Dataset 1 com 120 classes. (b) Dataset 2 com 83 classes.

Figura 1: Distribuição da quantidade exemplos de cada uma das classes de cada *dataset*.

B. Pré-Processamento dos Dados

Após a obtenção dos *datasets* é necessário um pré-processamento das imagens, pois as redes neurais precisam ser alimentadas com imagens de dimensões iguais. Em seguida, a intensidade dos pixels das imagens também precisa ser reescalada do intervalo [0, 255] para [0.0, 1.0] para que .

Para o *Dataset* 1, além do que foi descrito acima, também é fundamental separar o conjunto de treinamento em dois, de forma que 20% constitua o conjunto de validação e 80% o conjunto de treinamento.

Ambos os *datasets* não possuem muitos exemplos para cada classe, portanto, a utilização da estratégia de *data augmentation* é essencial. Ela permite a diversificação das imagens disponíveis, de forma que aplica técnicas como rotação, e *flipping* horizontal e vertical da imagem. Além dessas duas, foram também aplicadas técnicas de deslocamento de largura e altura, bem como *zoom*, e faixa de cisalhamento.

C. Redes Neurais Convolucionais Pré-Treinadas

As redes pré-treinadas utilizadas para modelagem desse trabalho foram VGG16, VGG19, ResNet50, Inception V3 e Inception-ResNet V2. Cada uma delas foi testada para ver o desempenho do treino de cada dataset a partir da técnica de *transfer learning*. A técnica de *transfer learning* ou *aprendizado por referência* consiste em utilizar um modelo já desenvolvido como ponto inicial para o desenvolvimento de outra tarefa.

1) VGG16: A **VGG16** é uma rede robusta que conta com 10 camadas de convolução com Max Pooling a cada 2 ou 3 delas. Isso é seguido por 3 camadas *fully connected* e um *softmax* no final. Essa rede é conhecida por ter 92.7% no teste de acurácia da ImageNet e estar no top-5 nesse quesito.

Em relação ao problema de classificação deste trabalho, esta rede não apresentou resultados satisfatórios para nenhum dos datasets, não convergindo nestes casos.

2) VGG19: A **VGG19** é uma rede também robusta, baseada na VGG16. Ela possui 3 camadas a mais, porém sua acurácia é bem parecida com sua antecessora.

3) ResNet50: A **ResNet50** é uma rede residual de 50 camadas que procura resolver os problemas de saturação e degradação da acurácia e de dificuldade de treino de redes mais complexas. Nessa ideia, redes residuais, ao invés de tentar aprender as features de cada camada, tentam aprender os resíduos, fazendo com que o treinamento seja mais fácil e resolva os problemas citados anteriormente.

Em relação a classificação de raças, essa rede teve um desempenho bom ao ser treinada com o dataset 1 (III-A1), como será explicitado mais a frente neste mesmo trabalho.

4) Inception V3: A **Inception V3** é uma rede extremamente projetada que procura fazer um modelo mais largo, ao invés de mais profundo. Ela conta com várias melhorias em relação a sua antecessora V2.

5) Inception-ResNet V2: A **Inception-ResNet V2** é uma rede híbrida entre a ResNet e a Inception. Ela possui o custo computacional similar a de uma Inception V4, porém adiciona noções de redes residuais em sua arquitetura. No que tange o trabalho apresentado, essa rede foi capaz de grandes resultados em ambos *datasets* (III-A).

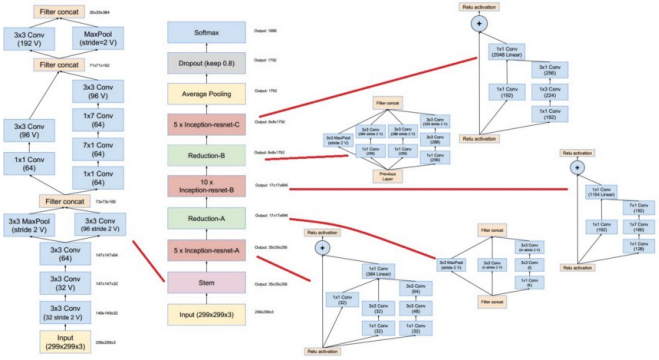


Figura 2: Arquitetura do modelo *Inception-ResNet V2*.

D. Camadas do Topo

As camadas do topo que substituíram as usuais de cada rede pré-treinada foram:

***GlobalAveragePooling2D*:** reduz a dimensão espacial de cada mapa de *features* de $h \times w \times d$ para $1 \times 1 \times d$, de forma que calcula a média de cada mapa.

***Flatten*:** converte o mapa de *features* em uma única coluna que é passada para a camada totalmente conectada.

***Dense*:** camada totalmente conectada à rede neural.

***Dropout*:** é responsável pela regularização, de forma que neurônios aleatoriamente selecionados são ignorados. Assim a rede se torna menos sensível à pesos específicos de alguns neurônios, o que constrói uma rede capaz de generalizar melhor e também diminui o *overfitting*.

ReLU *(Rectified Linear Unit)*: função de ativação que permite que o problema de classificação seja tratado como um problema não-linear, já que as imagens são compostas por diferentes objetos que não são lineares entre si.

***Softmax*:** função de ativação final que nos devolve a probabilidade de cada uma das raças de cachorro.

Foram testadas diversas variações de parâmetros das camadas, como o número das unidades na camada *Dense* e a taxa da camada de *Dropout*, além de intercalações entre a *GlobalAveragePooling2D* e a *Flatten*. Em geral, a ordem que se mostrou mais favorável para inserir as camadas do topo foi: *GlobalAveragePooling2D* ou *Flatten*, *Dense*, ReLU, *Dropout*, *Dense* (com o número de unidades fixo determinado pela quantidade de classes), *Softmax*.

E. Otimizadores

Como otimizadores, foram utilizados para avaliação os seguintes:

SGD *(Stochastic gradient descent)*: é um otimizador iterativo que escolhe amostras aleatóriamente para verificar o gradiente e fazer a otimização do algoritmo de aprendizado.

RMSprop *(Root Mean Square Propagation)*: nesse método, a taxa de aprendizado é adaptada para cada um dos parâmetros.

Adam *(Adaptive Moment Estimation)*: é uma atualização do RMSprop, utilizando-se não só do gradiente, mas do segundo momento do gradiente.

Os 3 foram testados após a escolha do modelo pré-treinado a ser utilizado para cada dataset para decidir qual possuia a melhor performance em cada caso.

IV. Resultados e Discussão

A. Dataset 1

Para o *dataset* 1, foram testados os modelos pré-treinados de VGG19, ResNet50, InceptionV3 e Inception-ResNet V2. É importante ressaltar que antes de diminuir a quantidade de classes desse *dataset* foram testados diversos modelos, configurações de camadas do topo, otimizadores e tamanho de entrada da imagem. Entretanto, havia a limitação da memória RAM de 12.72 GB, que impedia que fosse possível trabalhar com imagens maiores do que 96x96x3, provavelmente devido ao pré-processamento. Assim, obtinha-se *overfitting* no treinamento após atingir uma acurácia de validação de 0.5. Quando o *dataset* foi limitado para 35 classes foram obtidos resultados melhores, os quais serão discutidos em seguida.

Dentre os modelos citados, apenas a VGG19 não convergiu. Os treinamentos seguintes foram realizados a partir do modelo ResNet50, utilizando como otimizador RMSprop e Adam. É possível visualizar os resultados nas Figuras 3 e 4, respectivamente. Com o otimizador RMSprop com a taxa de aprendizado em 0.0001 e imagens de entrada de dimensões 224x224x3, as acurácias de validação e treinamento obtidas foram, respectivamente, 0.7095 e 0.7730. Já com o otimizador Adam com taxa de aprendizado em 0.00005 e imagens de entrada de dimensões 208x208x3 – houve a tentativa de utilizar dimensões de 224x224x3, porém a memória RAM não suportou –, as acurácias de validação e treinamento obtidas foram, respectivamente 0.8143 e 0.8167.

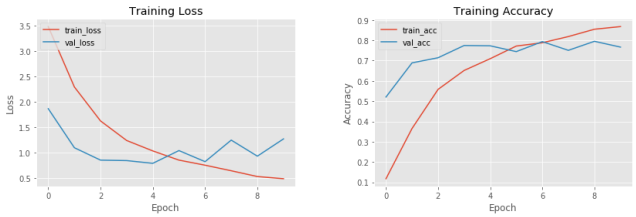


Figura 3: Figuras relacionadas ao treinamento do *dataset* 1 com a ResNet50 de modelo base e RMSprop como otimizador.

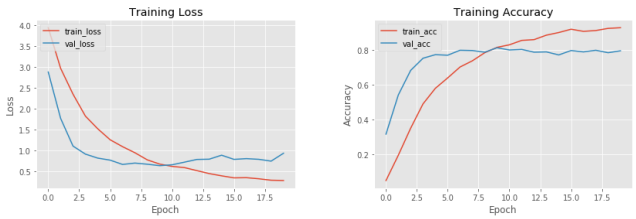


Figura 4: Figuras relacionadas ao treinamento do *dataset* 1 com a ResNet50 de modelo base e Adam como otimizador.

A partir do melhor resultado obtido com o otimizador Adam, o próximo passo era tentar outras variações de modelos base. A primeira tentativa foi utilizar o modelo InceptionV3 com as mesmas configurações citadas para o melhor resultado obtido com o modelo ResNet50. Entretanto, não houve aumento da acurácia de validação, e a acurácia de treinamento se mostrou inferior, aproximadamente 0.7952. Sendo assim, a segunda tentativa foi com o modelo Inception-ResNet V2, também com o otimizador Adam e taxa de aprendizado de 0.00005, mas com imagens de entrada de dimensões

199x199x3. Os resultados obtidos no treinamento podem ser vistos nos gráficos da Figura 5. A acurácia de validação do melhor modelo foi 0.8302 e a acurácia de treinamento foi 0.8393.

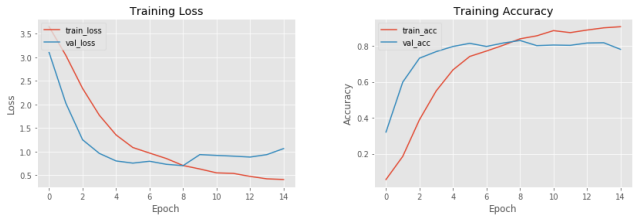


Figura 5: Figuras relacionadas ao treinamento do *dataset* 1 com uma Inception-Resnet V2 de modelo base e Adam como otimizador.

Os modelos finais de cada treinamento foram obtidos dada a *loss* de validação de menor valor, para que não fossem utilizados modelos com *overfitting* na apresentação dos resultados.

Na Figura 6, que representa a matriz de confusão obtida a partir da predição do conjunto de imagens de validação, é possível fazer uma análise melhor dos resultados do modelo utilizando Inception-ResNet V2. A classe que atingiu pior resultado na matriz foi a classe shih-tzu, que foi identificada em 9 imagens como lhasa. A Figura 7(e) mostra um exemplo de falha na predição da classe lhasa. Na imagem, ela foi classificada erroneamente como a classe shih-tzu. Entretanto, esse é um erro cometido por muitos humanos. As duas raças são realmente muito semelhantes no que diz respeito às características visuais. O terceiro maior caso de falhas é com a raça australian terrier, que foi classificada como cairn em 3 imagens, e como silky terrier em 4 imagens. Essas três raças também são comumente confundidas por humanos, por serem muito parecidas visualmente. Um exemplo da classe australian terrier classificada equivocadamente pode ser visto na Figura 7(d). O último caso de falha que analisaremos é o que o modelo classificou como beagle, quando na realidade o cachorro se tratava de um saluki. A imagem é a da Figura 7(f), e como pode ser visto, é uma imagem desafiadora, pois só é possível visualizar a cabeça do cachorro, que ocupa apenas um fragmento pequeno da foto. A olho nu, as características são relativamente semelhantes, dado que o beagle também tem orelhas marrons, e parte da região do focinho branca.

Por outro lado, temos um caso de acerto da raça bernese mountain dog, ilustrado na Figura 7(a), em uma imagem com uma certa dificuldade para classificação, pois parte do cachorro está ocluso por uma mesa no canto da foto, e em seu plano de fundo há pessoas em volta, bem como móveis. Essa raça, segundo a matriz de confusão, foi classificada muitas vezes de forma correta. A raça que não obteve nenhuma falha de predição no conjunto de validação avaliado foi a newfoundland, cujo exemplo pode ser visto na Figura 7(b). Por fim, o último caso de sucesso é apresentado na Figura 7(c), um pinscher bem reconhecível na imagem. Apesar dessa classe ter sido classificada erroneamente seis vezes, é possível perceber que com uma imagem de reconhecimento sem muitos empecilhos, como no caso da Figura 7(f), o modelo consegue performar satisfatoriamente.

Infelizmente, não foi possível analisar como o modelo final se comportou com as imagens do conjunto de teste, pois esse não estava anotado.

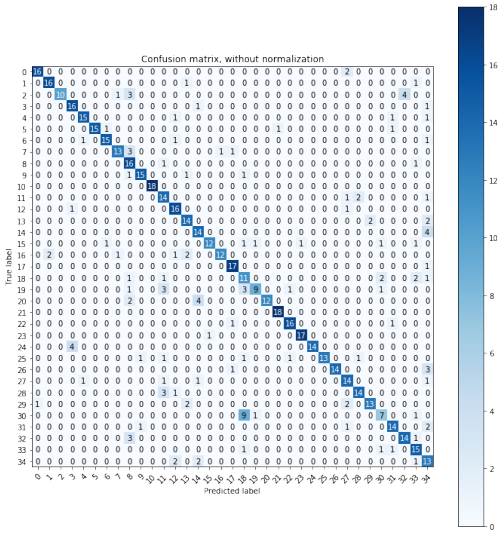


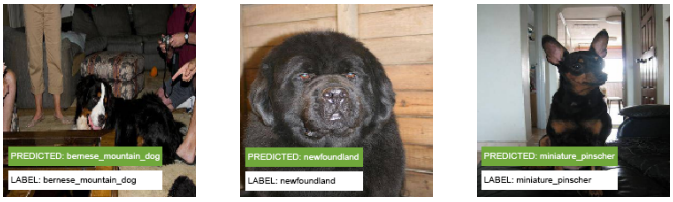
Figura 6: Matriz de confusão obtida após utilização do modelo final, que tem como modelo base o Inception-ResNet V2, para o *dataset* 1. Cada número na matriz representa uma classe, com a seguinte correspondência: (0: afghan hound), (1: airedale), (2: australian terrier), (3: basenji), (4: beagle), (5: bernese mountain dog), (6: blenheim spaniel), (7: border terrier), (8: cairn), (9: chow), (10: entlebucher), (11: great pyrenees), (12: ibizan hound), (13: irish wolfhound), (14: italian greyhound), (15: japanese spaniel), (16: lakeland terrier), (17: leonberg), (18: lhasa), (19: maltese dog), (20: miniature pinscher), (21: newfoundland), (22: norwegian elkhound), (23: papillon), (24: pembroke), (25: pomeranian), (26: pug), (27: saluki), (28: samoyed), (29: scottish deerhound), (30: shih-tzu), (31: siberian husky), (32: silky terrier), (33: tibetan terrier), (34: whippet)

B. Dataset 2

Para o *dataset 2*, foram testados os modelos pré-treinados de VGG16, ResNet50 e Inception-Resnet V2. Todos os modelos foram rodados com imagens 299x299 e com otimizadores com taxa de aprendizado de 0.0001. A implementação para o processamento das imagens nesse caso foi diferente para o dataset 1, feito durante o treinamento e não antes. Logo, a utilização de imagens maiores não foi prejudicada pela limitação da RAM.

Para os modelos testados, apenas o Inception-Resnet V2 convergiu. Podemos ver os resultados de acurácia e de perdas tanto da ResNet quanto da Inception-Resnet V2 nas figuras 8 e 9

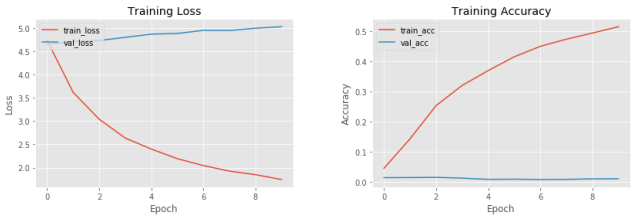
É importante perceber que no caso da Inception-Resnet V2, o gráfico de acurária mostra uma acurácia alta desde o



((a)) ((b)) ((c))

((d)) ((e)) ((f))

Figura 7: Resultados obtidos após a predição de algumas imagens do conjunto de validação do *dataset* 1, utilizando o modelo final que tem como modelo base o Inception-ResNet V2.



(a) (b)

Figura 8: Figuras relacionadas ao treino do dataset 2 com uma ResNet 50 de modelo base e SGD como otimizador

início do treino. Isso mostra que as camadas manualmente adicionadas não surtiram tanto efeito, dado que o modelo pré-treinado puro já era o suficiente para atingir resultados bons.

Depois da escolha do modelo pré-treinado mais eficiente, foi possível testar o impacto de cada otimizador dentro desse modelo. Nesse sentido, obtivemos os resultados mostrados na tabela I

Otimizador	Acurácia de treino	Acurácia de validação
SGD	0.822771	0.935071
RMSprop	0.776506	0.932248
Adam	0.784216	0.935237

Tabela I: Acurácia do modelo usando Inception-Resnet V2 para cada otimizador com taxa de aprendizado de 0.0001.

É possível ver alguns erros e acertos desse modelo nas figuras 10 e 11

Com base nas figuras, é possível dizer que o modelo teve mais dificuldade em classificar fotos em que os cachorros estavam com pessoas em primeiro plano na foto. Também, algumas fotos com poses não convencionais que fossem capaz de esconder algumas características importantes das raças foi

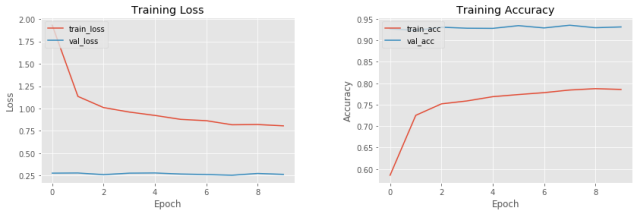
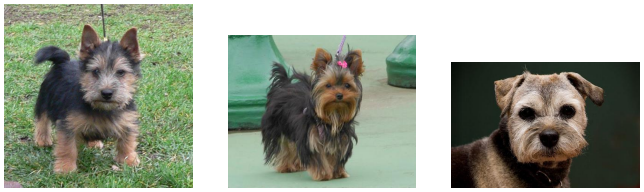


Figura 9: Figuras relacionadas ao treino do dataset 2 com uma Inception-Resnet V2 de modelo base e Adam como otimizador



(a) Norwich Terrier (b) Yorkshire Terrier (c) Border Terrier

Figura 10: Figuras em que o modelo conseguiu predizer a raça corretamente

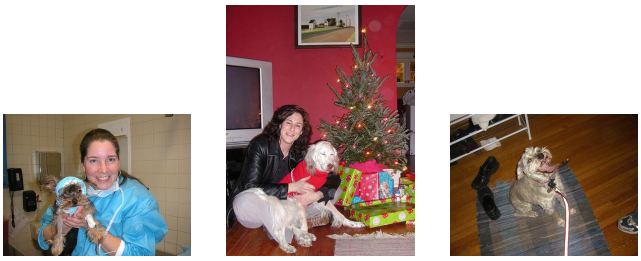
prejudicial para a acurácia. Para esse dois casos, os 3 exemplos encaixam: enquanto a figura 11b possui uma pessoa na foto e a figura 11c possui um cachorro numa pose não convencional, a foto 11a possui os dois problemas, dado que o cachorro está debilitado, enfaixado e com outra pessoa na fotografia.

V. Conclusões

A solução que apresentou maior valor de acurácia para ambos os datasets foi a que associou InceptionResNetV2 ao otimizador Adam. Por outro lado, o dataset 2 apresentou um valor de acurácia maior o que pode estar associado ao fato de utilizar as imagens com dimensão maior. Isso foi possível pela forma que a implementação foi realizada, processando as imagens no momento do treinamento. Pois, para o dataset 1, houve um pré-processamento das imagens de entrada o que demandou alto custo computacional e, para que funcionasse, foi necessário reduzir a dimensão das imagens.

Um dos principais problemas encontrados pelo grupo durante a codificação do projeto foi a limitação de memória RAM imposta pela ferramenta utilizada para treinamento. Por esse motivo, foi necessário reduzir o tamanho das imagens de entrada o que ocasionou uma diminuição na acurácia. Além disso, por se tratar de milhares de fotos e, sendo imagem um tipo de dado que necessita de muita mémoria, sua manipulação apresentou certa dificuldade.

A primeira abordagem do problema consistiu na busca por utilizar todas as classes do dataset 1 para a solução final, o que foi impossibilitado em decorrência do custo computacional que processar essa quantidade de imagens demandava. Dessa forma, a única maneira de não ultrapassar a memória RAM disponível foi diminuindo a dimensão das imagens, porém, com imagens menores não foi possível convergir e obteve-se uma baixa acurácia. E assim, houve a necessidade de diminuir



(a) Yorkshire Terrier - Lhasa Apso (b) English Setter - Clumber Spaniel (c)

Figura 11: Figuras em que o modelo não conseguiu predizer a raça corretamente no formato: raça real - predição

para 35 classes com o intuito de manter uma dimensão razoável para as fotos.

Os passos futuros desse projeto seriam aumentar a quantidade de classes, realizar testes com outras CNNs pré-treinadas, como a GoogLeNet, por exemplo, para poder realizar comparação entre mais redes e assim tentar chegar a um valor de acurácia maior. Outro possível passo futuro seria realizar o balanceamento do dataset 1 e analisar de que forma isso influenciaria nos resultados.

REFERÊNCIAS

[1] E. L. Colombini, "Introdução ao aprendizado de máquina." [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/Aula14_0.pdf 1

[2] ——, "Fundamentos de aprendizado de máquina." [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/Aula14_1.pdf 1

[3] ——, "Aprendizado de máquina." [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/Aula14_2.pdf 1

[4] ——, "Redes neurais." [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/Aula16.pdf 1

[5] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., ser. Series in Artificial Intelligence. Prentice Hall, 2010. [Online]. Available: http://aima.cs.berkeley.edu/ 1

[6] "Dataset kaggle." [Online]. Available: https://www.kaggle.com/c/dog-breed-identification/ 1, 2

[7] W. LaRow, B. Mittl, and V. Singh, "Dog breed identification," Ph.D. dissertation, Stanford University, 2016. 1

[8] Y. Aussat, "Dog breed identification," Ph.D. dissertation, University of Waterloo, 2017. 1

[9] A. Ayanzadeh and S. Vahidnia, "Modified deep neural networks for dog breeds identication," Ph.D. dissertation, Istanbul Technical University, 2018. 1

[10] E. L. Colombini, *Projeto 4*. [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/P4.pdf 1

[11] "Python 3." [Online]. Available: https://www.python.org/ 1

[12] "Keras." [Online]. Available: https://keras.io/ 1

[13] "Scikit-learn." [Online]. Available: https://scikit-learn.org/stable/ 1

[14] "Numpy." [Online]. Available: https://www.numpy.org/ 1

[15] "Pandas." [Online]. Available: https://pandas.pydata.org/ 1

[16] "Dataset 2." [Online]. Available: http://www.recod.ic.unicamp.br/~feandalo/MO444_dogs.tar.gz 2