Predizendo raça de cachorros mediante análise de imagens

Eduardo Rocha de Andrade
Stephane de Freitas Schwarz

1 INTRODUÇÃO

Com o avanço dos recursos computacionais e tecnológicos, diversas tarefas de importância prática foram beneficiadas, como por exemplo, a identificação de características biométricas de animais domésticos perdidos, mais especificamente cachorros. Nesse contexto, o principal objetivo desse trabalho é identificar a raça de cachorros mediante o processamento de imagens. Para tal, foram exploradas características extraídas por redes neurais pré-treinadas com o conjunto *ImageNet*. Os dados foram divididos em três grupos sendo treinamento, validação e teste, no qual contêm 83 classes ou raças.

Nas próximas seções foram abordados dois importantes passos da tarefa: extração de características, onde são apresentados alguns conceitos do processo adotado por algumas arquiteturas e o porquê apresentam resultados surpreendentemente bons, e a segunda parte, construção do modelo de classificação/*fine tuning*, quais foram os parâmetros que proporcionaram o melhor resultado e estratégias adotadas para evitar *overfitting*.

2 ARCABOUÇO PROPOSTO

O termo convolução é muito usado em visão computacional, e nos últimos anos tal técnica tem sido extensivamente integrada em redes neurais, devido sua alta eficiência. As convoluções além de reduzirem o número de parâmetros, são capazes de abstrair informações relevantes das amostras. Em visão computacional, as redes neurais convolucionais (CNN) podem ser aplicadas tanto para extração de características quanto para classificação de imagens.

Neste trabalho duas abordagens foram executadas. A primeira consiste em utilizar as redes somente na extração de *feature maps* e utilizar os mesmos em outros classificadores como SVM, *Random Forest* e Regressão Logistica. A segunda, consiste em substituir as últimas camadas não convolucionais da rede por outras mais específicas para este problema, permitindo um treinamento *end-to-end*. As métricas adotadas foram acurácia normalizada e *F1 score*.

3 ANÁLISE DE DADOS

Os dados são compostos por imagens de 83 raças de cachorros. Estes estão divididos em três grupos: treino, validação e teste, contendo respectivamente 8300, 6022 e 5420 imagens de diferentes tamanhos e formato *.jpg*.

Considerando-se que o conjunto de dados de treino possui um baixo número de imagens, é aconselhável a utilização de redes pré-treinadas em um maior número de dados. Desta maneira, foram escolhidas redes treinadas na competição *ImageNet* que possui 1000 categorias, incluindo algumas raças de cachorros.

Além de utilizar redes pré-treinadas, é aconselhável a utilização de *data augmentation*, que consiste em aplicar transformações nas imagens originais, e.g. rotações, espelhamento, *zoom*, cisalhamento, a fim de obter um conjunto de treino mais robusto, permitindo que a rede generalize com mais facilidade.

4 EXTRAÇÃO DE CARACTERÍSTICAS

Sabendo que o resultado da saída de cada camada da CNN é um vetor de características que representa uma abstração da imagem de entrada, utilizou-se o resultado do penúltimo nível, visto que é o mais representativo, como entrada para os classificadores SVM, *Randon Forest* (RF) e Regressão Logística (RL). Como pode ser visto na Tabela 1.

Tabela 1- Desempenho dos classificadores quando treinados com os mapas de características obtidos da última camada da CNN.

	SVM	RF	RL
Acurácia	0,513	0,514	0,712
F1	0,510	0,511	0,701

Como observado na Tabela 1, a performance dos classificados foi ruim, mesmo alterando os coeficientes de regularização, taxa de

aprendizagem e gama (nos casos que se aplicam). Uma explicação para esse infortúnio é a excessiva quantidade de características por imagem, o que ocasiona em pouca representatividade entre categorias. Com o propósito de evitar tal impasse foi utilizado a técnica de *data augmentation.* Dessa forma para cada imagem do conjunto de treinamento foram gerados amostras rotacionadas, invertidas e reescaladas. Tal abordagem também foi feita no conjunto de validação, no entanto houve uma melhora pouco significativa.

Nesse contexto, a mesma abordagem foi executada com a arquitetura ResNet50, isso porque o vetor resultante da última camada tem dimensões igual a (1, 1, 2048), desse modo é esperado uma melhora significativa no resultado das predições, visto que a quantidade de parâmetros é menor.

Tabela 2- Desempenho dos classificadores quando treinados com os mapas de características obtidos da última camada da CNN Resnet50.

	SVM	RF	RL
Acurácia	0,310	0,313	0,311
F1	0,303	0,309	0,311

Os resultados da Tabela 2 mostram que não houve uma melhora do desempenho do classificador com menos características por imagem, invalidando a hipótese anterior.

5 EXTRAÇÃO DE CARACTERÍSTICAS E CLASSIFICAÇÃO *END-TO-END*

Uma possível abordagem consiste em usar redes pré-treinadas, livrando-se das camadas *fully-connected* originais e então, adicionar novas camadas específicas para o problema em questão. Desta forma, os pesos das camadas convolucionais (extração de características) são àqueles pré-treinados na competição *ImageNet,* enquanto que os pesos das demais camadas (camadas de classificação) são inicializadas do "zero" (neste trabalhou utilizou-se o esquema de inicialização Xavier-Glorot [1]).

O treino da rede, por sua vez, foi divido em duas etapas. A primeira consiste em congelar os pesos das camadas convolucionais e treinar apenas as camadas de classificação. Tal procedimento é aconselhável pois evita que camadas já treinadas e recém-inicializadas sejam treinadas simultaneamente com a mesma *learning rate.*

A segunda etapa, conhecida como *fine-tunning*, consistem em treinar todas as camadas da rede com uma *learning rate* consideravelmente reduzida. Assim, é possível que os pesos pré-treinados das camadas convolucionais adaptem-se especificamente para o problema em questão.

5.1 Data Augmentation

Dois esquemas de *data augmentation* foram aplicados nesta tarefa. A primeira, referida por moderada, consiste em rotações, translações verticais/horizontais e espelhamento horizontal.

Já a segunda, consiste em transformações mais agressivas, a fim de evitar ainda mais o *overfitting*. Esta se utiliza de rotações, translações verticais/horizontais, *zoom*, cisalhamento e espelhamento tanto horizontal quanto vertical.

As transformações utilizadas em ambos os esquemas e seus respectivos valores estão dispostos na Tabela 3.

Tabela 3- Transformações utilizadas

Transformação	Moderada	Agressiva
Rotação	20°	360°
Translação vertical	0,2	0,4
Translação horizontal	0,2	0,4
Zoom	0	0,3
Cisalhamento	0	0,4
Espelhamento horizontal	Sim	Sim
Espelhamento vertical	Não	Sim

5.2 Implementação

A implementação foi feita em *Python* utilizando as bibliotecas *Keras*, *OpenCV* e *sklearn.* Devido a restrições de memória RAM, as imagens foram carregadas e processadas (*augmentation)* dinamicamente durante o treino em forma de lotes. Tal processamento foi paralelizado em diversos núcleos da CPU, enquanto que o treinamento ocorria na GPU. Desta forma, evitou-se eventuais gargalos devido à leitura/processamento das imagens durante o treino.

Na Figura 1 é possível observar o esquema utilizado na implementação. As camadas da rede são explicadas na próxima secção.

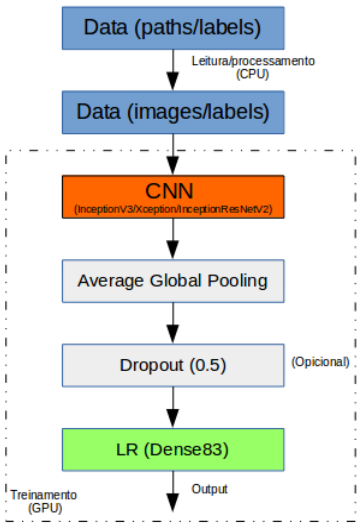

Figura 1- Esquemático utilizado. Azul: Pipeline de dados; Laranja: Estrutura da rede convolucional; Azul claro: Etapas adicionais; Verde: Camada de classificação

5.3 Resultados iniciais

A partir de testes inicias, percebeu-se que não era necessário a utilização de camadas do tipo *fully connected*. Acredita-se que tal fato é devido a alta capacidade entropica das redes utilizadas, uma vez que adicionando tais camadas resulta diretamente em um aumento considerável de *overfit.*

Nota-se na Figura o uso de *Global Average Pooling.* Apesar de não mandatório, o uso dessa camada no lugar de achatar os *feature maps* (*Flatten)* diminui consideravelmente o número de parâmetros da rede, sem comprometer a performance da mesma.

Diversas redes foram testadas a fim de obter o melhor resultado possível. As quatro melhores redes e seus respectivos parâmetros estão dispostas na Tabela 4. Em uma das redes testadas foi empregado o uso de uma camada de *Dropout* de 50% a fim de reduzir o *overfit* obtido.

O treinamento foi realizado em lotes de 10 imagens e consistiu nas duas etapas mencionadas na Seção 5. Nota-se que todas as redes foram treinadas usando método de optimização *ADAM* com *learning rate* de 0,001 para primeira etapa de treinamento (CNN congelada) e 0,00001 para a segunda etapa (*fine-tunning*).

Tabela 4- Redes utilizadas e seus respectivos parâmetros e resultados.

Arquitetura	Esquema de *augmentation*	*Dropout*	Acurácia obtida
InceptionV3	Agressiva	Sim (50%)	89%
Xception	Moderada	Não	90%
InceptionResNetV2	Moderada	Não	91%
InceptionResNetV2	Agressiva	Não	92%

5.4 *Test Time Augmentation* (TTA)

Uma possível maneira de melhorar os resultados obtidos na Seção 5.3 consiste em aplicar transformações nas imagens também durante a inferência. Desta forma, em vez de obter apenas uma predição para cada imagem, diversas predições são realizadas, uma para cada alteração da imagem original. Finalmente, os resultados são combinados de alguma maneira.

Um total de cinco tipos de transformações foram utilizadas, nos quais três são rotações: 90, 180 e 270° e as outras duas consistem em espelhamento (vertical e horizontal).

A combinação dos resultados foi feita utilizando três métodos distintos. O primeiro, *soft-voting*, consiste em somar as probabilidades de cada predição, levando em conta o grau de "certeza" de cada uma.

O segundo método foi um classificador de regressão logística (LR) treinado para escolher a melhor predição. Semelhantemente, o terceiro foi um classificador SVM com *kernel* RBF.

Infelizmente, tanto *soft-voting* quanto LR levaram a resultados piores que a predição original. Já o classificador SVM obteve uma melhora insignificante (+0,2%) comparado com o tempo necessário para execução da técnica. Desta forma, abriu-se mão da utilização de TTA nesta tarefa.

5.5 Ensembles

Outra técnica possível para melhorar os resultados, consiste em combinar as predições das quatro redes da Tabela 4.

Similarmente a Seção 5.4, foram utilizados três métodos para combinar os resultados: *soft-*

voting, classificador de regressão logistica e SVM (RBF).

Em contraste aos resultados obtidos ao utilizar TTA, a combinação das quatro redes resultou em uma melhora considerável (+ ~1%) para os três métodos utilizados, resultando em 93% de acurácia no conjunto de validação.

5.6 Resultados finais no conjunto de teste

Uma vez definido o método a ser utilizado, é possível retreinar os modelos nos conjuntos de treino mais validação e verificar a performance no conjunto de teste. Assim, aumenta-se consideravelmente o número de imagens disponíveis para treino.

Os resultados obtidos no conjunto de treino estão dispostos na Tabela 5. Nota-se uma melhora significativa em relação aos resultados da Tabela 4, fruto do aumento considerável do número de imagens de treino.

Tabela 5- Resultados obtidos no conjunto de teste. As redes foram retreinadas no conjunto (treino + validação).

Arquitetura	Esquema de *augmentation*	*Dropout*	Acurácia obtida **(teste)**
InceptionV3	Agressiva	Sim (50%)	91,2%
Xception	Moderada	Não	92,8%
InceptionResNetV2	Moderada	Não	91,8%
InceptionResNetV2	Agressiva	Não	93,3%

As quatro redes mais uma vez foram combinadas formando um *Ensamble,* resultando em uma melhora significativa de performance. Os resultados podem ser visualizados na Tabela 6.

Tabela 6- Resultados obtidos através da combinação das quatro redes.

Método	Acurácia obtida **(teste)**
Soft-voting	95,18%
Logistic Regressor	94,88%
SVM (RBF)	95,12%

Também foi utilizada a métrica conhecida como *F1-score,* que mede o compromisso entre *precision* e *recall,* para avaliar os resultados. Como tal métrica é originalmente definida para classificadores binários, foi utilizada a média ponderada (proporcional a quantidade de cada classe) da métrica *F1-score* de cada classe. O resultado foi um valor de **95,07%**.

6 CONCLUSÃO

Considerando os resultados obtidos na tarefa proposta, fica evidente o poder das redes neurais convolucionais. Além disso, ressalva-se a importância de *transfer learning,* uma vez que sem o uso de uma rede pré-treinada, seria praticamente impossível obter resultados semelhantes aos apresentados aqui, dada a atual conjectura: recursos computacionais limitados e um conjunto de treino pequeno.

Durante essa tarefa, foram utilizados métodos visando a melhoria dos resultados, independente dos custos computacionais dos mesmos. Entretanto, para certas aplicações é necessário que os modelos não exijam tanta memória ou poder computacional, como, por exemplo, aplicações para celular.

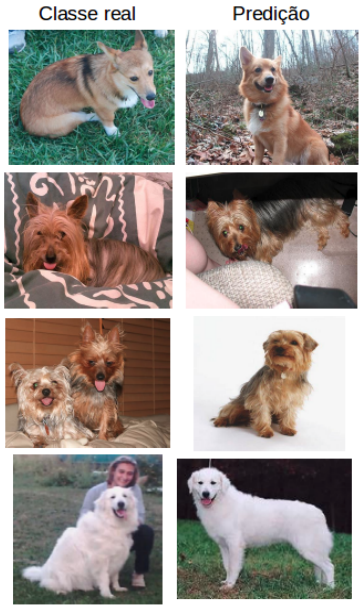

Figura 2- Exemplo das quatro classes mais confundidas. A coluna da esquerda representa a classe real, enquanto a coluna da direita representa a classe predita pelo modelo.

Por último, foi feita a matriz de confusão das predições realizadas pelo *Ensemble.* Nota-se através da mesma que apenas quatro combinações resultaram em mais de 10 erros. Exemplos destas classes podem ser vistas na Figura 2.

7 REFERÊNCIAS

[1] Glorot, X. & Bengio, Y.. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, in PMLR* 9:249-256