Estudo de Técnicas de Deep Learning para Reconhecimento de Espécies de Pássaros

Thauany Moedano - RA 92486
Departamento de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos, São Paulo
Email: t.moedano@unifesp.br

***Resumo*—Um dos trabalhos no ramo da biologia é identificar espécies de pássaros. Técnicas de aprendizado de máquina podem ajudar a automatizar esse processo, treinando um agente com uma base de dados existente fazendo com que o agente tenha conhecimento suficiente para distinguir e classificar espécies diferentes. Deep Learning é uma abordagem bem sucedida para classificação de imagens pois cria uma rede com muitas camadas ocultas, gerando uma grande quantidade de parâmetros para a tarefa de rotulação. Este trabalho utiliza diferenças configurações de redes para rotular espécies de pássaros**

I. Introdução/Motivação

Um trabalho pouco conhecido e árduo no ramo da biologia é o monitoramento de florestas. Esse trabalho consiste em observar a fauna e flora existentes em uma região e acompanhar a abundância ou ausência de animais, crescimento de filhotes, aparecimento de novas espécies entre outras informações que possam indicar alterações na fauna local.

Para realizar essa tarefa, geralmente se envia um grupo de biólogos para uma pesquisa de campo onde se captam sons e tiram fotos para posteriormente analisar os dados e classificar as espécies de acordo com um conhecimento prévio. Por exemplo, espécies do pássaro mosquiteiro podem ser diferenciadas pelas cores de suas plumas e pela frequência de seu canto [1]. Já zebras podem ser diferenciadas pelos padrões de suas listras [2].

O grande problema é que executar essa tarefa de reconhecimento manualmente pode causar muitos erros caso as pessoas designadas para a tarefa não tenham pleno conhecimento em como distinguir as espécies. A automatização desse trabalho torna o processo mais simples e rápido.

Utilizar fotos é uma das maneiras mais simples para reconhecer espécies. A partir de uma base de dados anotada é possível reconhecer diferentes espécies sem a necessidade de refazer o trabalho em campo. Um tipo de técnica que pode processar essa informação é a abordagem por *Deep Learning*. Essa abordagem consiste em utilizar redes neurais com muitas camadas ocultas, de modo que possa ser extraídos milhares de parâmetros para classificar uma imagem. Por produzir muitos parâmetros, os métodos que utilizam *Deep Learning* geralmente tem grande taxa de acerto [3].

Este projeto mostra como reconhecer espécies de pássaros utilizando a abordagem *Deep Lerning*.

II. Objetivos

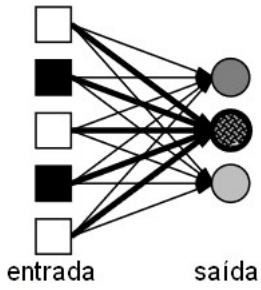
- Estudar como funcionam as técnicas de *Deep Learning*.
- Estudar diferentes modelos de rede neurais convolucionais.
- Analisar e discutir sobre a acurácia do modelo utilizado para reconhecer diferentes espécies de pássaros através de imagens.

III. Revisão Bibliográfica

A. Redes Neurais Artificiais

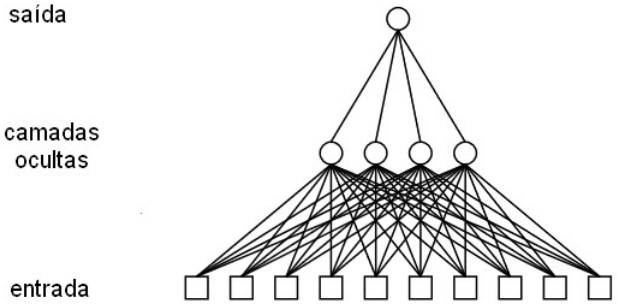
As Redes Neurais Artificiais são um tipo de aprendizado de máquina que se inspira em como o cérebro humano funciona. A ideia é modelar uma rede de circuitos que possa se ajustar a um conjunto de dados e executar uma classificação. Para isso, utiliza-se unidades chamadas neurônios em que recebem pesos utilizados em uma função de ativação e seu resultado é atribuído em uma saída. A rede neural mais simples apresenta uma camada de entrada e uma camada de saída e é conhecida como perceptron:

Fig. 1. Representação de uma rede neural com uma única camada [4]



Com o objetivo de otimizar as redes neurais, criaram-se camadas intermediárias entre entrada e saída. Essas camadas são conhecidas como camadas ocultas e potencializam o poder de aprendizado da rede neural.

Fig. 2. Representação de uma rede neural com múltiplas camadas [4]

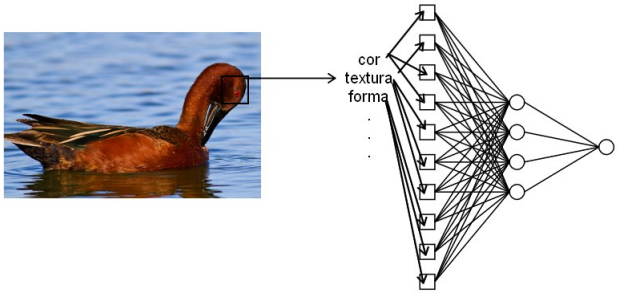


Embora múltiplas camadas otimizem uma rede neural, ainda é necessário ajustar os pesos manualmente. Esse problema é resolvido introduzindo a retropropagação de erro. Com isso, os pesos são atualizados automaticamente e a rede consegue ajustar os pesos da melhor forma sem nenhuma interferência manual.

B. Deep Learning

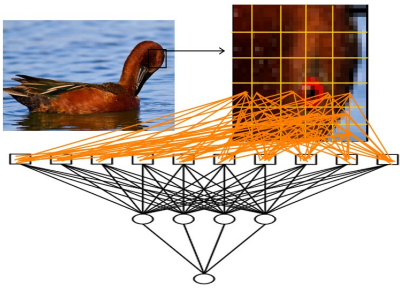
Na abordagem tradicional, as características são extraídas da imagem e servem como entrada de uma rede neural, por exemplo.

Fig. 3. Funcionamento tradicional de uma rede neural articificial



Com *Deep Learning*, o foco é criar muitas representações para alimentar muitas camadas ocultas. A imagem é divida em várias partes menores e a entrada é alimentada por pixels brutos. O número de parâmetros criados é muito maior do que em uma rede convencional.

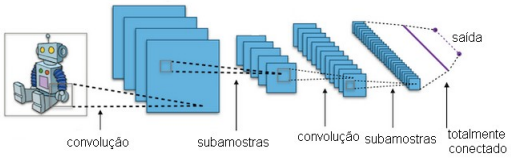
Fig. 4. Funcionamento de uma rede neural utilizando *Deep Learning*



C. Redes Neurais Convolucionais

Outro artifício que *Deep Learning* usa para extrair os parâmetros são as redes neurais convolucionais. Essas redes possuem uma estrutura específica na qual seleciona uma parte da imagem e convoluciona para um subconjunto de amostras [5]. Cada subconjunto representa uma camada de convolução. As convoluções geram uma saída para a camada oculta.

Fig. 5. Exemplo de convoluções [6]



IV. Metodologia Experimental

A. Base de Dados

O primeiro passo é ter uma grande base de dados anotada para executar a tarefa de aprendizado. A base escolhida chama-se *NA Birds* [7] e conta com 70.000 imagens rotulando cerca de 400 espécies de pássaros. A base também divide algumas espécies de pássaros por fases da vida: filhote, jovem ou adulto.

Como os recursos para processamento eram limitados, foram escolhidos apenas algumas espécies de toda a base para realizar os experimentos. 3 conjuntos de dados foram definidos para os testes.

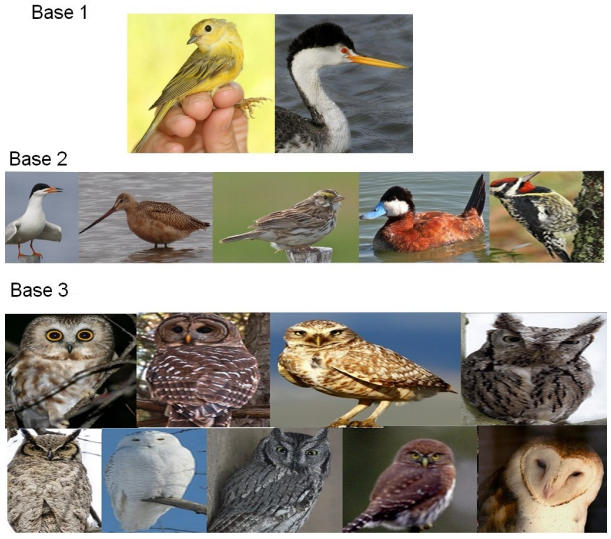
1) Conjunto 1 - Ou um ou outro: Esse conjunto contém duas espécies de pássaros: *Mergulhão de Pescoço Preto*, um pássaro conhecido pela capacidade de ”andar” sobre a água e *Mariquita Amarela*.

2) Conjunto 2 - Cinco pássaros diferentes: Esse conjunto contém mais classes de pássaros distintos. Entretanto, algumas similaridades podem ser encontradas como o tamanho, o local onde as fotos foram tiradas, bico e formato. A lista contém o *Pato De Rabo Alçado Americano*, cujo costuma aparecer em desenhos animados antigos sempre sendo caçado, a *Gaivina de Forster*, cujo se assemelha com uma andorinha, *Maçarico-marmóreo*, *Pica-pau De Barriga Amarela* e o *Tico-tico dos Prados*.

3) Conjunto 3 - Somos todos corujas: Esse conjunto testa a capacidade de *Deep Learning* apurar espécies muito parecidas. Aqui todos os pássaros são de diferentes espécies de corujas. A lista contém: *Coruja das Torres*, *Coruja das Torres do Oeste*, *Coruja das Torres do Leste*, *Corujão Orelhudo*, *Coruja das Neves*, mesma espécie da coruja que aparece no filme *Harry Potter*, *Corujinha Caburé*, *Coruja Buraqueira*, *Coruja Barrada* e *Coruja Serra Afiada*.

Cada classe de espécies teve seu conjunto de imagens dividido para treino e teste. O único processamento feito sobre as imagens foi uma reescala para atender os padrões da biblioteca utilizada, como será explicado a seguir.

Fig. 6. Representação de cada uma das bases de dados



B. Softwares utilizados

Os experimentos foram executados utilizando as bibliotecas *Keras* [8] e *Theano* [9]. São bibliotecas em *Python* onde é possível construir modelos de redes de maneira bem simples.

O primeiro passo foi importar para o programa as imagens a serem utilizadas.*Keras* possui um pacote chamado *ImageGenerator* onde é possível tratar as imagens de maneira simples. Para isso duas pastas, uma para teste e uma para treino, foram criadas e, dentro de cada pasta existem subpastas contendo as imagens de cada classe. A biblioteca é capaz de reconhecer essa estrutura de arquivos e importar as imagens para serem utilizadas.

Após isso, basta construir o modelo de rede neural a ser utilizado. *Keras* tem uma classe de modelo que, com funções simples, é possível adicionar novas camadas à rede. A seguir, uma breve explicação de cada camada que foi utilizada na construção do modelo:

- Conv2D: É uma camada de convolução convencional. Seus parâmetros são: número de filtros que definem a dimensionalidade da saída; tamanho do *kernel*, que especifica o tamanho da janela de convolução; função de ativação; e o formato da entrada, que especifica a dimensão das imagens e o canal de cores.
- ZeroPadding2D: Também é uma camada de convolução mas adiciona zeros nas bordas da imagem. Seu único parâmetro define a quantidade de pixels a serem adicionados. Isso faz com que as convoluções continuem gerando um grande número de parâmetros.
- MaxPooling2D: É uma camada de *pooling*. Serve para concatenar atributos vizinhos. Seu único parâmetro é o tamanho da margem de *pooling*.
- Dropout: Essa função trata de desativar aleatoriamente alguns neurônios a fim de encontrar a melhor combinação de parâmetros. *Dropout* só é habilitada durante o treino. Seu parâmetro define a porcentagem de chance de um neurônio ser desativado.
- Flatten: Essa camada é responsável por normalizar as saídas, concatenando toda a informação em um único vetor.
- Dense: Última camada do modelo, representa a camada totalmente conectada. Seus parâmetros são o número de classes e uma função de ativação.

Keras também fornece funções para salvar o modelo em imagem e sumarizá-lo na tela, mostrando a quantidade final de parâmetros gerados.

Para treinar o modelo é necessário antes compilá-lo, ou seja, definir um otimizador, uma métrica de validação e uma função de perda. Após compilar o modelo, já é possível treiná-lo utilizando a função *fit generator* que divide o conjunto de imagens em ciclos de treino. É possível salvar os pesos utilizando a função *save weights*. Após treinar o modelo basta usar a função *evalute generator*. No final são obtidos dois resultados: O valor da função de perda e a acurácia.

C. Definindo um modelo de rede neural

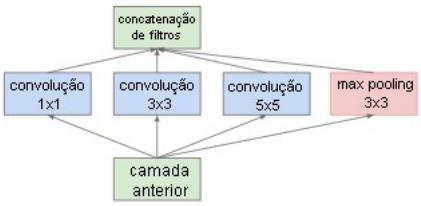
Na literatura existem alguns modelos de redes neurais já definidos e testados. Entretanto, esses modelos geralmente são voltados para trabalhar em bases extremamente grandes.

O exemplo mais famoso de arquitetura de redes neurais é a AlexNet [10] que possui cinco camadas convolucionais e três camadas totalmente conectadas. A função de *MaxPooling* é aplicada em três das cinco camadas. Isso gera cerca de 60 milhões de parâmetros que são usados para rotular imagens em 1000 classes.

A VGGNet [11] é outro exemplo de modelo que aposta na profundidade. Possui 19 camadas convolucionais onde todas possuem a janela de convolução 3x3. Apesar de ser profunda, é uma rede bastante simples.

Já a Google foi na contramão da ideia de simplicidade e apresentou a GoogLeNet [12]. A GoogLeNet possui 22 camadas mas em cada camada ocorre uma série de operações em paralelo.

Fig. 7. Exemplo de uma camada da GoogLeNet



Seguindo a ideia de profundidade, surge a Microsoft ResNet [13], uma rede neural de 152 camadas. Nesse tipo de rede, para uma entrada x é feita uma série de convoluções. Ao final, encontra-se uma função h(x). Em redes tradicionais, dizemos que h(x) é a função equivalente a f(x) (a função que verdadeiramente representa nosso conjunto de saída). Entretanto, a ResNet soma a função encontrada a entrada original. Desse modo, quando ocorre a retropropagação, a

atualização dos pesos é mais acertiva pois está calculando em cima de diferenças residuais.

Todas as redes apresentadas anteriormente tem um alto grau de acurácia. Entretanto, para atingir essa acurácia é necessário uma base de dados enorme. Todas as redes foram treinadas em bases de dados com cerca de 15 milhões de imagens, o que justifica a quantidade de camadas e parâmetros gerados por essas redes.

No caso desse trabalho tem-se apenas no máximo 9 classes contendo 872 imagens para treinar e testar. Utilizar uma arquitetura complexa resultaria em *overfitting*, ajuste demasiado sobre o treino. Para evitar *overfitting* e garantir uma boa acurácia, a estratégia utilizada foi criar uma versão simples e similar a *AlexNet* e a *VGGNet*.

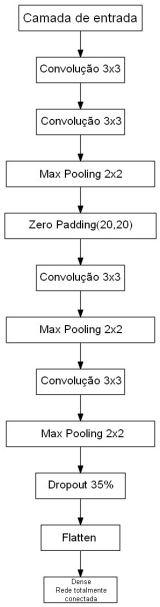
O modelo construído possui quatro camadas de convolução sendo que em três ocorrem *pooling* e em uma aplica-se *zero padding*. Além disso, experimentou-se utilizar a função de dropout com intuito de evitar *overfitting*.

Todas as camadas de convolução tem como função de ativação a função ReLU. A função ReLU é definida por: $f(x) = max(0, x)$ onde x é entrada de um neurônio. Somente na última camada totalmente conectada utilizou-se a função de sigmóide.

Para a função de perda utilizou a função *Cross Entropy* definida como: $L_i = -\sum_y y_{i,j} log p_{i,j}$, onde y é a saída e p é a predição. O otimizador para atualização de pesos utilizado foi o *Adam*, um algoritmo que implementa a técnica gradiente-descendente.

Todas as funções relatadas acima foram utilizadas pelas redes citadas anteriormente. Por fim, essa é a estrutura final da rede construída nesse experimento:

Fig. 8. Modelo final da rede neural convolucional



V. Resultados e Discussões

A. Treino do modelo sugerido

A rede apresentada anteriormente foi utilizada para treinar e validar o três conjuntos de dados. Os resultados estão na tabela a seguir:

TABELA I
RESULTADO DA EXECUÇÃO DA REDE NEURAL PARA OS TRÊS CONJUNTOS DE DADOS

Conjunto	Parâmetros gerados	Acurácia no treino	Acurácia no teste	Perda
1	98.338	98%	94%	0.34
2	202.885	86%	74%	0.87
3	342.281	79%	69%	1.52

Para o conjunto de dados 1, o modelo não teve grandes problemas em aprender a diferenciar as duas classes. Como as duas classes eram de espécies muito distintas, em habitats distintos, a tarefa de aprendizado para a rede foi fácil.

No caso do segundo conjunto de dados, a acurácia caiu em 20%. E para o terceiro conjunto de dados os acertos são menores ainda, caindo mais 5% em relação ao segundo conjunto. A seguir, uma lista de possíveis causas para a perda de acurácia:

- Ruídos no conjunto de dados: Algumas das imagens utilizadas não mostravam nitidamente o pássaro. Existem exemplos em que galhos de árvores estão na frente dos pássaros ou a imagem é de baixa resolução. Além disso, há pássaros que podem ter exemplares de cores diferentes, fazendo parecer duas espécies distintas.

Fig. 9. Exemplo de pássaros de mesma espécie com cores diferentes e imagem que não mostra o pássaro nitidamente

- Poucos exemplos: Como já dito anteriormente, o sucesso de *Deep Learning* está relacionamento a gerar muitos parâmetros para uma enorme massa de dados. Nesse caso, existia poucos exemplares por espécies e talvez a rede não tenha conseguido descrever de maneira suficiente os dados observados.
- Modelo não gera bons parâmetros: Também há a possibilidade de que o modelo construído não esteja gerando

bons parâmetros. Existem muitos recursos que podem ser utilizados para alterar um modelo de rede neural convolucional e encontrar um que gere bons parâmetros não é uma tarefa trivial. Encontrar um bom modelo em *Deep Learning* ainda parece uma tarefa mais empírica do que teórica embora as últimas arquiteturas de redes neurais de sucesso estejam utilizando alguns novos conceitos teóricos.

VI. Conclusão

Com o crescimento do aprendizado de máquina, é possível substituir o trabalho manual de reconhecimento de espécies de pássaros por um processo automatizado e inteligente. A abordagem por *Deep Learning* é uma das mais bem sucedidas maneiras de se fazer classificação. Entretanto, construir bons modelos de redes neurais não é uma tarefa trivial e ainda envolve em grande parte um processo empírico. Mas se tiver em mãos uma grande base de dados livre de ruídos e bem anotada e um bom modelo de arquitetura de rede neural, é possível obter grandes taxas de acerto na tarefa de classificação.

VII. Trabalhos Futuros

Existe a necessidade de melhor analisar a base de dados e utilizar técnicas para aumentar os conjuntos de dados (por exemplo, girar as imagens, dar zoom e fazer recortes). Além disso, é necessário encontrar uma melhor maneira de separar as espécies a fim de evitar ruídos. Por exemplo, as espécies de pássaros com cores diferentes poderiam ficar em classes distintas.

Obtendo uma base de dados grande e mais acertiva, pode-se testar arquiteturas de modelos previamente treinados como a *VGGNet*. Ainda deseja-se executar a tarefa de classificação sobre todas as classes disponíveis na base *NABirds* utilizando uma GPU para agilizar a tarefa.

Agradecimentos

Os dados da base *NABirds* foram fornecidos pela *Cornell Lab of Ornithology* com agradecimentos aos fotógrafos e contribuidores da fundação *AllAboutBirds.com*. O material é baseado em um trabalho patrocinado pela *National Science Foundation*.

Referências

[1] J. A. C. Uy, R. G. Moyle, and C. E. Filardi, "Plumage and song differences mediate species recognition between incipient flycatcher species of the solomon islands," *Evolution*, vol. 63, no. 1, pp. 153–164, 2009.

[2] V. C. Couldridge and G. J. Alexander, "Color patterns and species recognition in four closely related species of lake malawi cichlid," *Behavioral Ecology*, vol. 13, no. 1, pp. 59–64, 2002.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2013.

[5] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition." in *Interspeech*, 2013, pp. 3366–3370.

[6] "Keras tutorial: The ultimate beginner's guide to deep learning in python," https://elitedatascience.com/keras-tutorial-deep-learning-in-python, acesso em 15/06/2017.

[7] A. Poole, "The birds of north america online. cornell lab of ornithology, ithaca," 2013.

[8] "Keras," https://keras.io, acesso em 15/06/2017.

[9] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[13] K. Zhang, M. Sun, X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.