PD5 - Reconhecimento de Objetos

Frederico Guth (18/0081641)
Tópicos em Sistemas de Computação, ,
Turma TC - Visão Computacional (PPGI)
Universidade de Brasília
Brasília, Brasil
fredguth@fredguth.com

***Resumo*—Detecção de pele lida com o reconhecimento dos pixels que representam pele em uma dada imagem. Cor é frequentemente usada por ser invariante a orientacao, tamanho e ser fácil de processar. Neste projeto, detectamos cor de pele usando tres tipos de classificadores: por limiarizacao, bayesiano simples e k-nn. Os resultados comparam qualidade da segmentação em relação ao tempo de execução dos algoritmos.**

***Index Terms*—deteção de pele, espaço de cores, K-NN, naive bayesian, threshold-based**

I. Introdução

Reconhecimento de Objetos lida com a identificação de objetos em imagens. Quando humanos vêem uma imagem, podem facilmente identificar objetos, pessoas, lugares;algo tão natural para nós é uma tarefa bastante complexa para um algoritmo e, até pouco tempo, com resultados nada animadores.

O momento crucial no crescimento meteórico do interesse por *deep learning* se deu em 2012, justamente na maior competição de reconhecimento de objetos, a ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [4]. O time liderado por Alex Krizhevsky foi o primeiro a usar redes neurais convolucionais profundas (RCPs) na competição e ganharam por larga margem [1]. Desde então, técnicas baseadas em RCPs tem sido as mais bem sucedidas para este problema.

A. Objetivo

O objetivo deste projeto é propor e avaliar um método de reconhecimento de objetos para a base de imagens CalTech101 [2].

II. Revisão Teórica

A. Redes Neurais Convolucionais Profundas

Redes Neurais Convolucionais são modelos computacionais inspirados na biologia do cortex visual. O cortex visual tem pequenas regiões de células sensíveis a regiões específicas do campo visual [3]. Da mesma forma, nas RCPs filtros são convolucionados em camadas gerando novas camadas que representam onde o filtro ”sensibilizou”a entrada.

A ideia é resolver o problema da representação do conhecimento introduzindo representações que são expressas em termos de outras representações mais simples [4]. Em essência, um RCP é apenas uma função matemática que mapeia um conjunto de valores de entrada a valores de saída, formada pela composição de várias funções mais simples. Com um

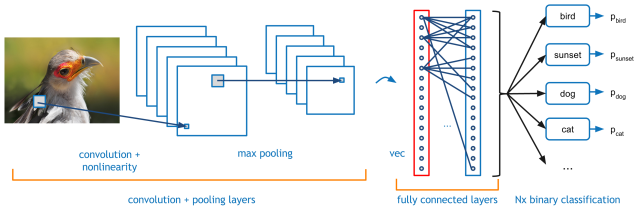

Figura 1: Uma rede neural convolucional profunda

número suficiente de composições, camadas, é possível obter funções de alta complexidade [3], [4].

Em tarefas de classificação, as camadas amplificam características da entrada que são importantes para discriminação das amostras e suprime variações irrelevantes. Uma imagem, entra como um tensor de valores de pixel, a primeira camada tipicamente representam a presença ou ausência de bordas em determinadas orientações e localizações na imagem. A segunda, detecta padões simples e arranjos de bordas. A terceira pode compor os padrões simples em combinações maiores que correspondem com partes de objetos, as camadas subsequentes irão detectar objetos como combinações dessas partes [3].

B. Transferência de Conhecimento

Em nosso dia a dia, transferimos conhecimento a todo momento. Aprender a tocar piano, facilita aprender tocar órgão. Reconhecer maçãs talvez ajude a reconhecer peras. Pessoas conseguem inteligentemente aplicar conhecimento prévio para resolver novos problemas com maior eficácia e eficiência [5] E algoritmos?

Pesquisa em transferêcia de conhecimento tem atraído mais e mais atenção desde 1995, quando foi tema de um workshop na NIPS-95 que discutiu a necessidade de métodos de aprendizado de máquina que retém e reusam conhecimento previamente obtido [5].

No contexto de RCPs para reconhecimento visual de objetos, fica claro que as camadas iniciais: capazes de reconhecer bordas, padrões e partes de objetos, treinadas com um conjunto de imagens para reconhecer determinados rótulos, pode ser usada em outro conjunto de imagens totalmente diferente ainda que os rótulos também sejam diferentes. A capacidade de utilizar os pesos resultantes de treinamentos com milhões de

imagens representa uma grande economia de processamento e uma ferramenta muito útil.

C. Hiperparâmetros e Ajuste fino

O aspecto fundamental do *deep learning* é que as camadas de características não são extraídas manualmente por pessoas; elas são aprendidas a partir dos dados usando procedimentos de aprendizados genéricos. Como consequência, RCPs podem ser retreinadas para diferentes tarefas de reconhecimento e classificação, permitindo-se aproveitar redes pré-existentes.

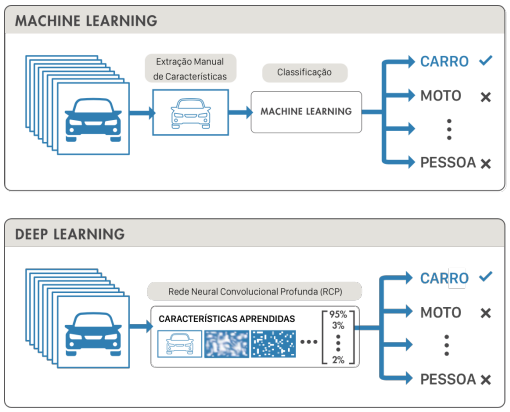

Figura 2: Machine Learning versus Deep Learning

Para isso, entretanto, é preciso ajustar a rede para o problema em questão. Esse ajuste é obtido variando hiperparâmetros: learning rate, número de épocas, tamanho do *batch*, função de ativação, inicialização, *dropout*, etc. De certa forma, pode-se pensar que até a arquitetura utilizada (ResNet, Inception, LeNet, etc) é um parâmetro.

É interessante notar que *deep learning* é epistemologicamente muito mais próximo das ciências naturais do que do resto da Ciência da Computação. Em deep learning o resultado empírico é crucial, o ajuste de hiperparâmetros pode ter tanto ou mais valor do que o desenvolvimento de novas arquiteturas e a criatividade em pensar formas de visualizar o resultados pode levar a novos insights. Sendo Ciência da Computação acostumada a presar o método dedutivo antes de tudo, tais características geram desconfiança e estranheza. Por outro lado, abre-se caminho para novas pessoas, com outras formas de pensar.

D. A Base Caltech101

Caltech 101 foi compilada em 2003 por Fei-Fei Li, Marco Andreetto, Marc 'Aurelio Ranzato e Pietro Perona no California Institute of Technology [2], com o propósito de ser utilizada para problemas de reconhecimento de objetos em imagens. A base contém 9146 imagens, divididas em 102 categorias, 101 de objetos e uma de background.

Já é uma base antiga e algumas características a tornavam boas para aquele momento, como:

- pouca variação de tamanho e posição do objeto de interesse;
- pouca oclusão
- pequeno número de categorias

Hoje, entretanto, tais características são vistas como fraquezas da mesma, uma vez que na prática, imagens como a da Caltech101 são incomuns.

Além disso, a base conta com um número bastante limitado de categorias e imagens, algumas categorias contendo apenas 31 imagens. Não sendo possível treinar com mais do que 30 imagens.

III. Método

A. Materiais

Foram utilizados:

- Servidor Paperspace/Fastai: GPU 8GB, 30GB RAM, 8 CPU
- Python 3.6.4 :: Anaconda custom (64-bit)
- Pytorch 0.3.0
- OpenCV 3.4.0
- 1 Notebook Jupyter
- 1 programa python equivalente ao Notebook.

Todos os arquivos do projeto estão publicamente disponíveis em git@github.com:fredguth/unb-cv-3183.git

B. Visão Geral

O método de reconhecimento de objetos desenvolvido nesse projeto foi fortemente influenciado por [6] e constitui-se das seguintes etapas:

1) Definir arquitetura RCP e obter rede pré-treinada com imagens da base ImageNet
2) Aumentar ao máximo a base de dados
   - Validação Cruzada
   - Imagens Criadas (Data Augmentation)
3) Otimização da Taxa de Aprendizado (Learning Rate).
4) Otimização em Tempo de Teste
   - Test-Time Augmentation
   - Test-Time model average

C. Arquitetura e Transferência de Aprendizado

A ResNet se provou bastante robusta para diversas tarefas de reconhecimento visual [7]. A motivação da sua criação veio do fato que com RCPs cada vez mais profundas surgiu um fenômeno de degradaçao do erro que não podia ser explicado por *overfitting*. A estratégia da ResNet para atacar o problema foi aumentar a profundidade da rede empilhando blocos de rede do mesmo tipo. Essa regra simplifica e reduz o número de hiperparâmetros e minimiza a degradação.

Neste projeto utilizamos a arquitetura de RCP ResNet50, pré-treinada com a base ImageNet. Uma das vantagens de usar a biblioteca fast.ai [6] é justamente a facilidade com que isso é feito.

Automaticamente, a biblioteca remove a camada final da Resnet-50 e adapta para o nosso problema onde a saída é a probabilidade de cada uma das 102 categorias. Mais especificamente, essa adaptação se dá com as seguintes camadas:

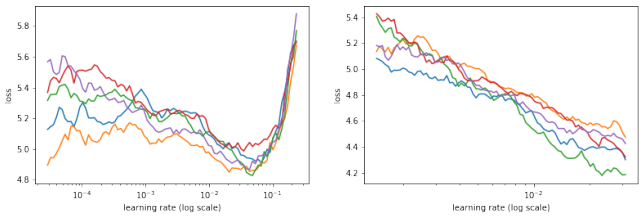

Figura 3: Otimização da Taxa de Aprendizado

D. Validação Cruzada

Validação Cruzada (Cross Validation) é uma técnica para criar uma base de validação a partir da base de treinamento. Em uma validação cruzada k-fold, a amostra original é aleatoriamente particionada em subamostra em k partes de igual tamanho, mutualmente exclusivas. Depois do particionamento, uma das subamostras é utilizada para teste e as demais k-1 para treinamento. Ao final de k interações temos 5 modelos treinados em bases diferentes. ¡imagem¿

E. Data Augmentation

Uma outra forma de contornar o problema de falta de dados é adcionar dados falsos no conjunto de treinamento com características que sabemos serem similares aos dados reais. Para diversos problemas gerar dados falsos não é uma abordagem viável. Mas em problemas visuais, sabemos que podemos aplicar operações de translação, rotação, escala e alteração cromática em imagens e melhorar os resultados do treinamento. ¡imagem¿

F. Otimização da Taxa de Aprendizado (Learning Rate)

Modelos de RCPs são normalmente treinados usando um otimizador por gradiente descendente estocástico (Stochastic Gradient Descent ou SGD). Há diversos tipos de otimizadores: Adam, RMSProp, Adagrad, etc. Todos permitem a definição da taxa de aprendizado (learning rate), que é o quanto o otimizador deve mover os pesos na direção do gradiente para um determinado mini-batch.

Com uma taxa pequena, o treinamento é mais confiável, mas exige mais tempo e processamento para chegar a um valor mínimo da função de perda. Se a taxa é maior, anda-se a ”passos mais largos”, mas o treinamento pode não convergir ou até divergir.

Para nos guiar na decisão da taxa mais adequada para o nosso problema, usamos a técnica descrita em [8]: começa-se a treinar a rede com uma taxa de aprendizado bem baixa e a aumentamos exponencialmente a cada batch.

Na figura **??**-a vemos que de início a função de perda melhora muito lentamente, e depois começa a acelerar a partir de 0.001, até que por volta de 0.1 a taxa de aprendizado se torna grande demais e o treinamento diverge, ou seja, a perda começa a aumentar.

A ideia é selecionar o ponto no gráfico com a perda mais rápida. No nosso exemplo, a perda diminui rapidamente na proximidade da taxa de aprendizado 0.012 (**??**-b), que é a que decidimos usar. Selecionar uma boa taxa inicial para o treinamento é apenas o começo, a ideia mais interessante da técnica é como se altera a taxa de treinamento durante o treinamento. O mais comum é definir uma taxa de decaimento, mas fugindo ao senso comum, [8] sugere uma variação cíclica em que a taxa de aprendizado pode sofrer aumentos repentinos. A figura **??** mostra um gráfico com a função de variação da taxa de decaimento que usamos em nossos treinamentos.

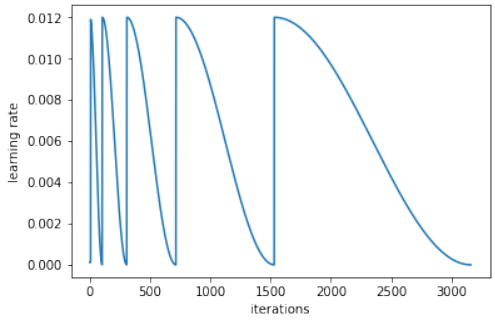

Figura 4: Mínimos Locais e Taxa de Aprendizado

A justificativa é facilmente entendida na figura xx do próprio artigo. A maneira convencional nos ajuda a chegar em um mínimo da função de perda que pode ser local. Já usando uma variação cíclica, podemos chegar a vários mínimos diferentes, permitindo-se até obter um mínimo global, uma ideia que, apesar do artigo não mencionar, remonta a outra mais antiga, Otimização por Recozimento Simulado (Simulated Annealing Optimization) [9].

IV. Resultados

Nesta seção apresentamos os resultados obtidos. Todos os dados podem ser acessados no repositório do projeto (III-A).

A. Segmentação usando subconjunto da SFA (10 imagens)

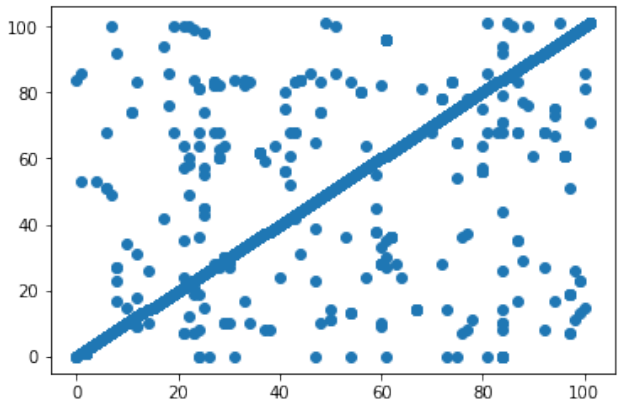

Figura 5: Resultado qualitativo dos diferentes algoritmos.

O resultado qualitativo obtido com os diferentes algoritmos pode ser apreciado em .

Tabela I: Base SFA 1118 imagens
(782 treinamento, 167 validação, 167 teste)

Algoritmo	Qualidade		Tempo Execução	
	Acuidade (%)	Jaccard (%)	Trein. (s)	Teste (s)
Nulo	79.59	0.0	0	0
Lim (B)	69.64	1.38	0	3.33
(CbCr)	95.03	77.78	0	3.01
Bayes	94.62	76.77	364	77
K-nn	94.49	77.00	406	1660

Para uma análise quantitativa da qualidade e do tempo de execução, temos I

B. Análise dos resultados

Em face aos resultados obtidos, cabem algumas análises:

1) O uso de um algoritmo nulo se mostrou muito importante, uma vez que apontou que a métrica escolhida como acuidade (pixels segmentados corretamente em relação ao total de pixels da imagem) é uma métrica muito ruim, uma vez que dado que as cores que representam pele são um pequeno cluster dentro do espaço de cores, $P(bg)$ é muito maior, logo o algoritmo nulo consegue bons resultados de acuidade.
   O índice Jaccard é melhor nesse sentido, mas não discrimina falsos positivos de falsos negativos.
2) O algoritmo por limiarização CbCr apresentou um resultado quantitativo melhor do que o esperado. Já esperava-se que fosse rápido, e foi, classificando 167 imagens de teste em apenas 3 segundos. Há de se considerar o fator ”sorte”na escolha dos limites para essa base. A análise qualitativa, entretanto, mostra que para amostras pequenas é razoavelmente pior do que o classificador bayesiano e o k-nn.
3) A limirização apenas pelo canal B em RGB tinha pouca chance de ir bem e confirmou o que esperávamos. Mais interessante foi notar que a biblioteca OpenCV [10] não apresenta nenhuma diferença de tempo para carregar imagens no espaço RGB (que é o mesmo da imagem de origem) e carregar no espaço YCbCr.
4) Os algoritmo bayesiano se mostrou uma alternativa mais robusta que o de limiarização, apresentando um resultado muito melhor na base pequena e bastante rápido também, levando 77 segundos para testar 167 imagens (menos de 0.5 segundo por imagem em uma CPU antiga).

V. Discussão e Conclusões

Neste trabalho, implementamos três algoritmos para a segmentação de cor de pele em imagens e apresentamos os diferentes custos-benefícios.Demonstramos que, em conformidade com a teoria existente, a segmentação de pele por cor usando algoritmos simples de classificação atinge bons resultados que são mais do que suficiente para diversas aplicações de pré-processamento.

Referências

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[2] F.-F. Li, R. Fergus, and P. Perona, “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories.” *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” vol. 521, pp. 436–44, 05 2015.

[4] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[5] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2009.191

[6] J. Howard *et al.*, “fastai,” https://github.com/fastai/fastai, 2018.

[7] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 5987–5995. [Online]. Available: https://doi.org/10.1109/CVPR.2017.634

[8] L. N. Smith, “Cyclical learning rates for training neural networks,” in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, mar 2017. [Online]. Available: https://doi.org/10.1109/wacv.2017.58

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: http://www.jstor.org/stable/1690046

[10] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.