

Chapter

1

Estudo Comparativo de Técnicas de Visão Computacional para Classificação de Frutas e Vegetais

Guilherme Henrique Galelli Christmann, Jacky Baltes e Rodrigo da Silva Guerra

Abstract

There is an increasing trend towards the use of Deep Learning techniques for the automation of various processes and systems. This work explores traditional techniques used in computer vision such as bag of features (BoF) models and Histograms of Oriented Gradients (HOG) compared with modern approaches based on Convolutional Neural Networks (CNNs). For this, we present two scenarios for fruit and vegetable classification from images. In one scenario, the objects are placed in a well-behaved environment with differences only in lighting and object position. The other scenario is more challenging and contains fewer images and much more background clutter. We show that, although most modern computer vision work is based on CNNs, traditional techniques can still achieve comparable results in some scenarios.

Resumo

Há uma tendência crescente no uso de técnicas de Deep Learning na automação de processos e sistemas. Este trabalho explora técnicas tradicionais de visão computacional como o modelo bag of features (BoF) e Histogramas de Gradientes Orientados (HOG), comparados com abordagens modernas baseadas em Redes Neurais de Convolução (CNNs). Para tal, definimos dois cenários para classificação de frutas e vegetais em imagens. Em um dos cenários, os objetos são colocados em um ambiente bem-comportado com diferenças apenas de iluminação e posição. O outro é um cenário mais desafiador, com menos imagens e maior presença de ruído no background. Apesar da maior parte dos trabalhos de visão computacional modernos basearem-se em CNNs, demonstramos que técnicas tradicionais podem alcançar resultados comparáveis.

1.1. Introdução

Grande parte dos esforços da área de *Machine Learning* (ML) são destinados à automação de processos e serviços tradicionalmente realizados por humanos. A evolução da capacidade de processamento nos últimos anos proporcionou o barateamento de CPUs e GPUs poderosas, estabelecendo um ambiente propício para o desenvolvimento, aprimoramento e uso de técnicas baseadas em ML na indústria. A aplicação de tais técnicas muitas vezes supera o desempenho de humanos e têm-se provado uma ferramenta valiosa para profissionais das mais diversas áreas. Seu potencial pode ser observado em aplicações como predição de câncer [Kourou et al. 2015], estudo de genética e do genoma humano [Libbrecht and Noble 2015] e análise de imagens de satélite [Jean et al. 2016], para citar algumas.

Técnicas de classificação e detecção de objetos em imagens têm-se tornado cada vez mais precisas, com uma preferência crescente ao uso de modelos baseados em Redes Neurais (RNs), mais especificamente Redes Neurais de Convolução (CNNs). Tradicionalmente, técnicas de visão computacional utilizam algoritmos específicos para extrair *features* das imagens, e.g. SIFT, ORB, SURF, etc. Esse tipo de *features* são chamadas de engenhadas, quando o desenvolvedor manualmente decide características relevantes ou utiliza algum processo específico para extraí-las. Exemplos de técnicas, que envolvem um passo de extração de *features* previamente à classificação, são o modelo *bag of features* (BoF) e histograma de gradientes orientados (HOG), que serão apresentados em detalhe neste trabalho. Em contrapartida, RNs e CNNs são capazes de automaticamente extrair *features* relevantes e realizar classificação no mesmo processo. Esse tipo de treinamento é chamado *end-to-end*, operando diretamente sobre os dados brutos, sem a necessidade de um método específico para extração de *features*. RNs modernas são compostas de milhões de parâmetros treináveis, distribuídos em camadas interconectadas, onde a saída de uma camada é conectada à camadas posteriores. Hardwares modernos permitem que RNs sejam construídas com diversas camadas de profundidade, resultando no chamado aprendizado profundo (*Deep Learning*). Essas múltiplas camadas permitem aprender representações com múltiplos níveis de abstração, descobrindo estruturas complexas nos dados [LeCun et al. 2015]. Entretanto, um fator que deve ser levado em consideração é o fato que RNs profundas necessitam de GPUs de alta performance para serem treinadas.

Um exemplo de um processo rotineiro que ainda não foi amplamente automatizado e será explorado neste trabalho é classificação automática de frutas e vegetais durante a pesagem em supermercados. Geralmente, a identificação ainda é feita por um humano, funcionário do estabelecimento que consulta uma tabela para determinar seu preço. Segundo [Rocha et al. 2010] ótimos resultados podem ser obtidos utilizando fusão de diferentes *features* engenhadas e extração de *background*. De maneira semelhante, [Zhang et al. 2014] aplicam PCA [Ke and Sukthankar 2004] após uma etapa de fusão de diferentes *features* engenhadas, reduzindo sua dimensionalidade e utilizando uma RN como classificador. De maneira *end-to-end*, [Zhang et al. 2017] utilizam uma CNN profunda de 13 camadas que opera diretamente sobre as imagens, juntamente com técnicas de aumento de dados (*Data Augmentation*) [Perez and Wang 2017]. Porém, outros resultados apresentados por [Akbari Fard et al. 2016], que utilizaram de CNNs rasas de apenas 3 camadas, sugerem que há uma necessidade de modelos profundos para atingir um bom desempenho.

Nesse trabalho, exploraremos tantas técnicas para classificação tradicionais baseadas em *features* engenhadas, extraídas com algoritmos especializados, e técnicas baseadas em CNN que operam diretamente sobre as imagens, de maneira *end-to-end*, extraíndo automaticamente as *features* e treinando o classificador em um único processo. Para tal, utilizaremos dois conjuntos de dados (*datasets*) distintos contendo imagens de frutas e vegetais: o *dataset* fornecido por [Rocha et al. 2010] com algumas modificações, e outro de nossa própria autoria com maior presença de *background clutter*.

1.2. Fundamentação Teórica

Classificação é uma das tarefas mais estudadas no campo de ML, e é definida como o problema de atribuir uma categoria à cada item de um conjunto de dados [Mohri et al. 2018]. Como afirmado anteriormente, o principal objetivo deste trabalho é comparar diferentes métodos de classificação baseado em imagens. Nesta seção, serão detalhadas as fundamentações teóricas de cada um dos métodos explorados neste trabalho.

1.2.1. *Bag of features* (BoF)

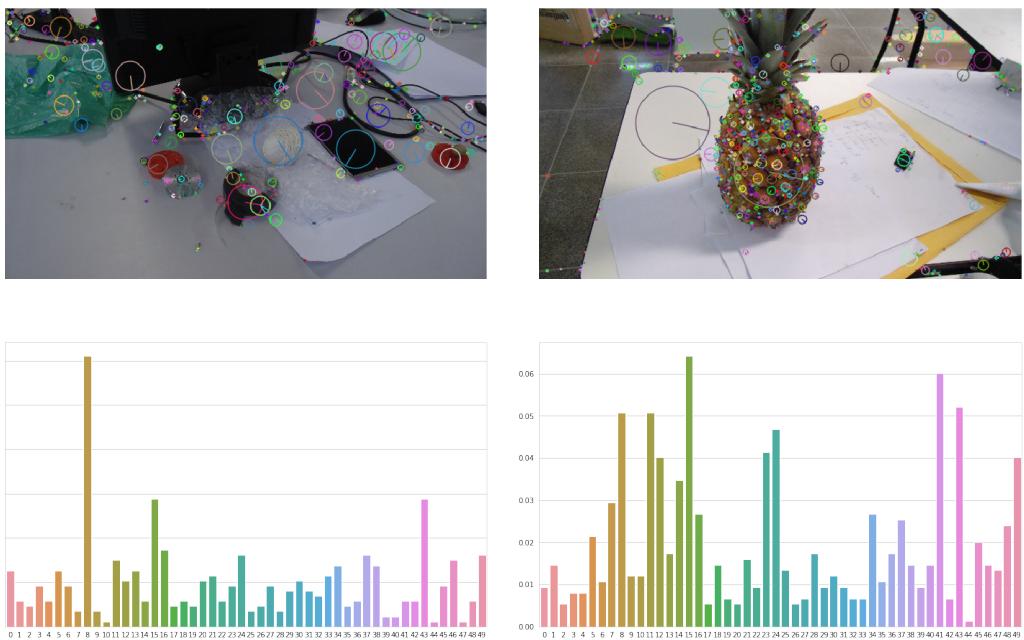
O BoF é uma técnica de visão computacional com raízes em uma técnica clássica para classificação de textos chamada de *Bag of Words* (BoW) [Sebastiani 2002]. No BoW a classificação é realizada a partir de uma representação alternativa do texto na forma de um histograma. Define-se um dicionário de palavras conhecidas, e o histograma é gerado contando o número de ocorrências de cada palavra do dicionário presente no texto.

O BoF é um análogo do BoW no contexto de visão computacional, onde o dicionário codifica características visuais das imagens, permitindo que uma imagem seja representada através de um histograma. Para construir o dicionário visual, primeiramente é preciso extrair *features*, características relevantes, das imagens. Neste trabalho, exploramos dois algoritmos populares para extração de *features*, que operam de forma semelhante: ORB [Rublee et al. 2011] e SIFT [Lowe 1999]. Basicamente, o processo é realizado em dois passos: (1) detectam-se pontos de interesse na imagem, regiões em que há grande variação de contraste e (2) gera-se um vetor de números reais para cada ponto de interesse detectado, a partir das características da imagem em sua vizinhança. Esses vetores são chamados descritores, e têm tamanho 32 para ORB e 128 para SIFT. Uma característica muito útil de ambos algoritmos é que seus descritores são invariantes à rotação, escala e diferenças de iluminação. O dicionário visual, então, é construído utilizando alguma técnica de agrupamento (*clustering*) nos vetores descritores. Em [Mohri et al. 2018], *clustering* é definido como o problema de particionar um conjunto de dados em subconjuntos homogêneos. Para esta tarefa, utilizamos o algoritmo *K-Means* [Duda et al. 2012].

Com o dicionário visual construído, é possível gerar uma representação alternativa das imagens, na forma de um histograma da ocorrência das características visuais presentes no dicionário. Para uma nova imagem, suas *features* são extraídas e seus descritores computados utilizando ORB ou SIFT. Para cada descritor, confere-se qual o grupo (*cluster*) correspondente no dicionário, incrementando um ocorrência para tal grupo. Ao final do processo, tem-se uma distribuição da ocorrência de cada palavra visual. A Figura 1.1 apresenta exemplos de imagens com os pontos de interesse SIFT detectados e o histograma

gerado a partir de seus descritores com um dicionário com 50 grupos. O fluxograma da Figura 1.2 apresenta o processo da construção do dicionário. Após, um classificador pode ser treinado sobre representações das imagens em forma de histograma. Neste trabalho, utilizamos uma Máquina de Vetores Suporte (SVM) [Burges 1998] como classificador a partir do histogramas do *bag of features*.

O modelo de *bag of features* ainda é objeto de pesquisa nos últimos anos. Em [Anthimopoulos et al. 2014] o método foi aplicado de maneira similar ao explorado neste trabalho, com extração de *features* SIFT, agrupamento por *K-Means* e um classificador SVM, em um cenário de classificação de comidas para diabéticos, alcançando bons resultados. Ainda mais recente, em [Minaee et al. 2017], histogramas de BoF foram utilizados em conjunto com outras *features* para determinar a ocorrência de lesões cerebrais em imagens de ressonância magnética.



(a) Foto sem nenhuma fruta ou vegetal e histograma correspondente

(b) Foto de um abacaxi e histograma correspondente

Figure 1.1: Imagens com pontos de interesse SIFT detectados e sua representação em histograma utilizando dicionário com 50 grupos.

1.2.2. Histogramas de Gradientes Orientados (HOG)

O uso de Histograma de Gradientes Orientados foi popularizado pelo trabalho de [Dalal and Triggs 2005], onde os autores demonstraram sua aplicação em detecção de pedestres, superando outras técnicas utilizadas até então. O método consiste em dividir a imagem em sub-regiões de mesmo tamanho, denominadas células, e computar um histograma a partir do gradiente da imagem para cada célula. Posteriormente, o histograma de cada célula é normalizado usando um número de células em sua vizinhança, chamada bloco. Os histogramas normalizados são concatenados em um longo vetor, gerando uma nova

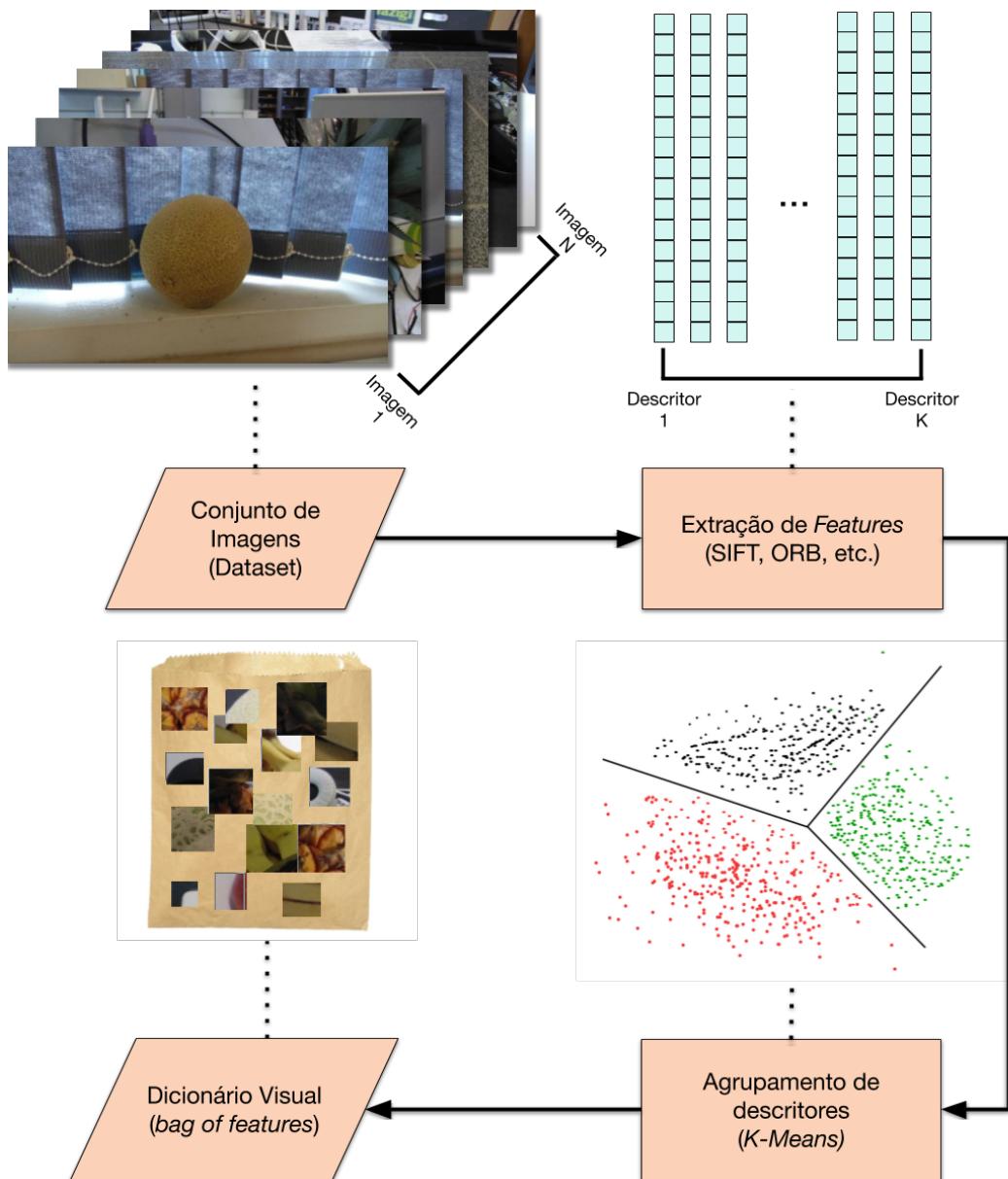


Figure 1.2: Fluxograma simplificado do processo de construção do dicionário visual: *bag of features*.

representação das características da imagem. Intuitivamente, a ideia é que a aparência local de um objeto juntamente com seu formato possam ser caracterizados pela distribuição dos gradientes de intensidade locais e a direção de arestas [Dalal and Triggs 2005]. O vetor resultante é chamado de descriptor HOG, e classificadores convencionais podem ser treinados para operar sobre os vetores. Neste trabalho, também utilizamos um classificador SVM para operar sobre esse assim como no modelo BoF, para fins de comparação. A Figura 1.3 apresenta um fluxograma simplificado do processo de computação do descriptor HOG.

O gradiente da imagem é computado com o operador *Sobel*, comumente utilizado como um algoritmo detector de arestas. O operador *Sobel* é uma maneira eficiente de aproximar o gradiente de uma imagem, e é realizado através de uma convolução entre a imagem e os *kernels* apresentados na Equação 1. O resultado pode ser observado no terceiro passo no fluxograma da Figura 1.3. Os gradientes horizontais são o resultado da operação com o *kernel* G_x e os verticais com G_y . A magnitude e a orientação do gradiente em cada *pixel* é determinada através das Equações 2 e 3, onde X_G é o resultado da convolução da imagem original com G_x , e Y_G o resultado da convolução com G_y . Após, cada célula gera um histograma com um número de *bins* N , onde cada *bin* representa uma direção distinta de 0° a 180° . Cada *pixel* da célula "vota" em um *bin* de acordo com sua orientação e magnitude. Os histogramas das células são normalizados de acordo com os valores mínimos e máximos dos histogramas pertencentes ao mesmo bloco. Ao fim, todos os histogramas são concatenados em um longo vetor, chamado descriptor HOG.

Assim como o modelo de *bag of features*, descritores HOG apresentam características úteis que continuam relevantes para aplicações modernas. [Tian et al. 2016] apresentaram duas extensões do descriptor HOG tradicional, e demonstraram sua eficiência na detecção de caracteres latinos, chineses e bengali. Outra aplicação interessante é detecção de imagens falsificadas, apresentada em [Lee et al. 2015]. Os autores utilizaram descritores HOG para detectar regiões e objetos duplicados nas imagens.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (1)$$

$$\text{Magnitude}(x, y) = \sqrt{X_G(x, y)^2 + Y_G(x, y)^2} \quad (2)$$

$$\text{Orientação}(x, y) = \text{atan2}(Y_G(x, y), X_G(x, y)) \quad (3)$$

1.2.3. Redes Neurais e CNNs

Grande parte do corpo de pesquisas modernas nas áreas de *Machine Learning* e Visão Computacional são compostas por métodos baseados em RNs e CNNs. Na realidade, as técnicas foram estabelecidas e tiveram suas primeiras aplicações nos anos 80 e 90. Por exemplo, a teoria apresentada em [Bishop et al. 1995] ainda é fundamental nas aplicações de hoje e, em sua maior parte permanece representativa das técnicas atuais. De fato, o que desencadeou a utilização em massa de tais modelos foi o crescimento exponencial da

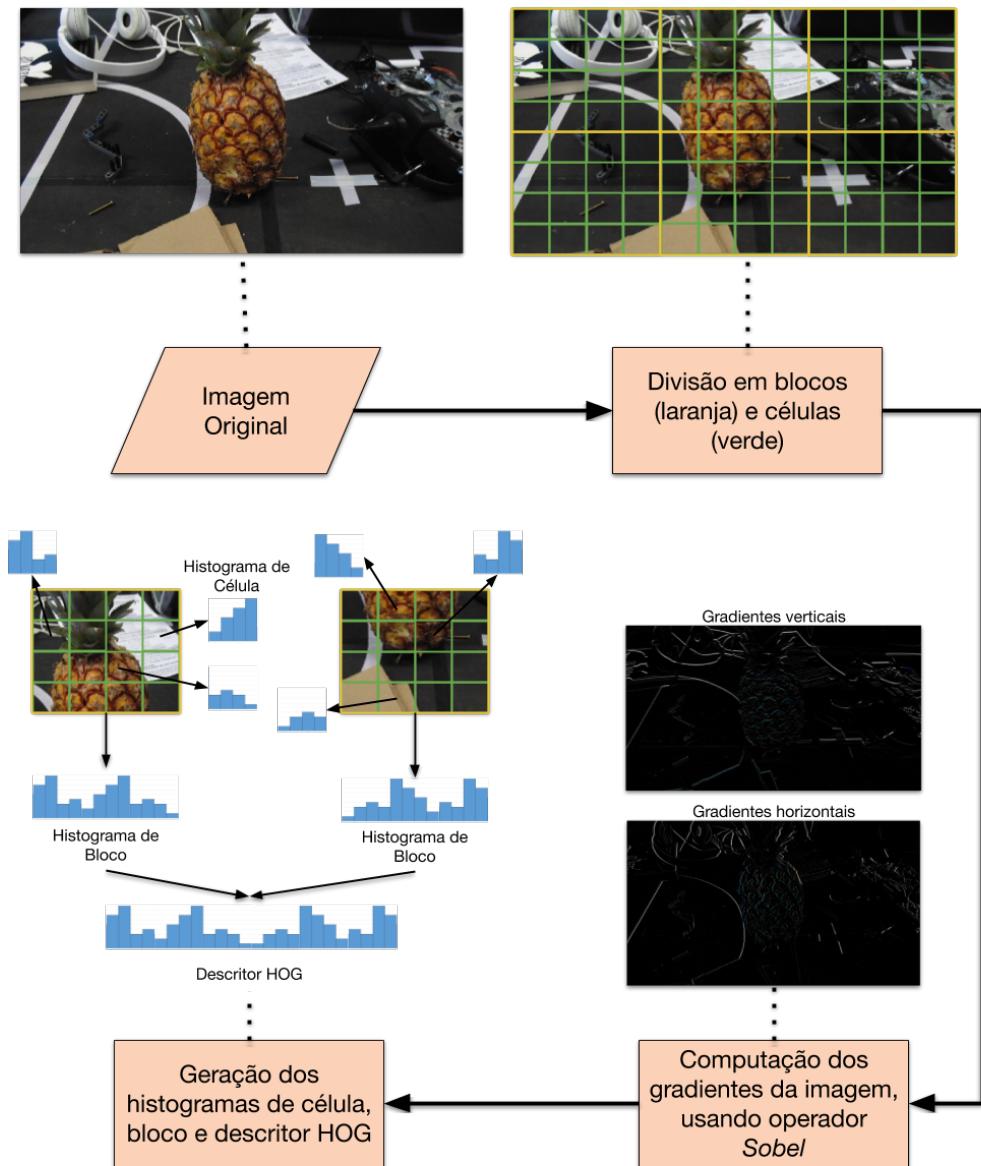


Figure 1.3: Fluxograma simplificado do processo de computação do descritor HOG de uma imagem.

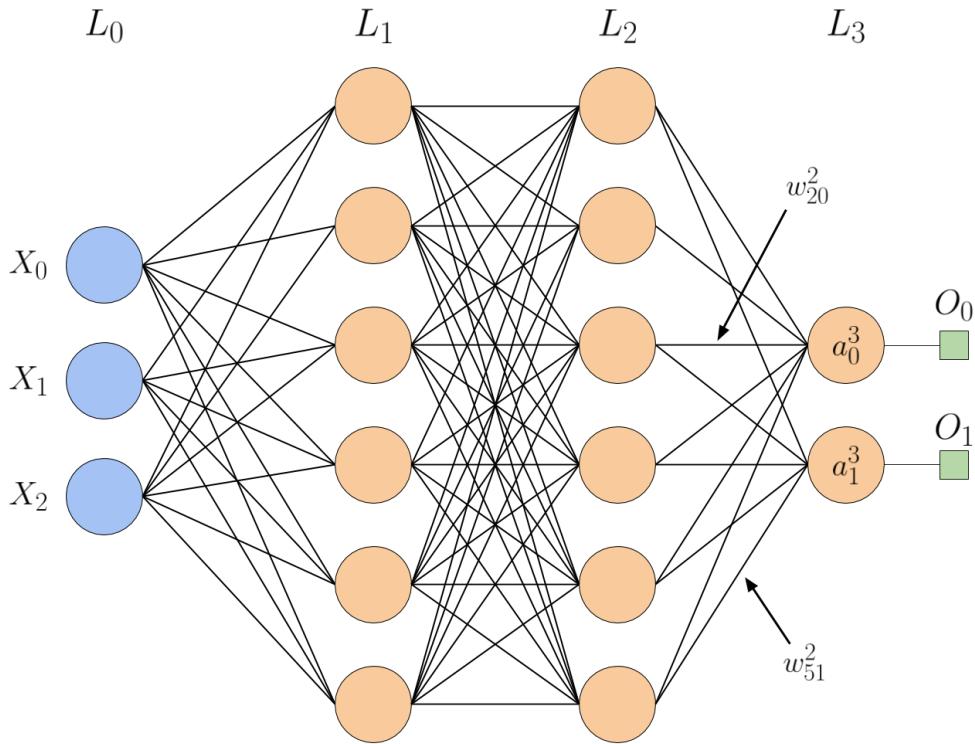


Figure 1.4: Exemplo de RN *feedforward* com 2 camadas escondidas.

capacidade de computação de GPUs, permitindo a criação de modelos mais complexos e sendo capazes do famoso aprendizado profundo (*Deep Learning*).

RNs são uma coleção de neurônios (também chamados de nós), comumente organizados em camadas interconectadas. Uma das configurações mais comuns é chamada de *feedforward*, onde cada neurônio de uma camada é conectado com todos os neurônios da próxima camada. A conexão entre dois neurônios possui um número real associado, chamado peso, que multiplica a saída do neurônio anterior e o resultado é utilizado como entrada do neurônio posterior. Os pesos são os parâmetros treináveis de uma RN, onde modelos complexos possuem um número na ordem dos milhões. A saída de um neurônio i em uma camada L é descrita pela seguinte equação:

$$a_i^L = \sigma \left(\sum_k w_{ki}^{L-1} \cdot a_k^{L-1} + b_i^L \right) \quad (4)$$

onde k são os neurônios da camada anterior, w_{ki}^{L-1} o peso associado da conexão do neurônio k ao neurônio i , e a_k^{L-1} é a saída do neurônio k da camada anterior. O termo b_i^L é um termo regularizador, chamado *bias*. O resultado dessa operação é passado por uma função σ , chamada função de ativação. A função σ é utilizada para controlar o comportamento da saída de um neurônio, e geralmente é do tipo não-linear. A Figura 1.4 apresenta um exemplo de um RN *feedforward* simples. O vetor de entrada $X = (X_0, X_1, \dots, X_n)$ e o vetor de saída $O = (O_0, O_1, \dots, O_n)$ são representados como camadas. As camadas intermediárias, entre a entrada e saída, são chamadas de camadas escondidas.

Redes Neurais de Convolução (CNNs) funcionam de maneira similar à RNs, com uma diferença importante: os pesos de uma CNN são os valores de um *kernel*. Cada camada de uma CNN possui N *kernels* (ou filtros), e gera como saída N novas imagens. Cada imagem é o resultado de uma operação de convolução das imagens de entrada da camada com cada um de seus *kernels*. As imagens saída de uma camada de convolução podem ser propagadas para outra camada, e assim por diante. Aumentando o número de camadas de RNs e CNNs é possível modelar padrões cada vez mais complexos a partir dos dados de entrada.

O treinamento de modelos de redes neurais é realizado através do algoritmo de *backpropagation* [Rumelhart et al. 1988]. A partir do vetor de entrada X e uma saída esperada O_e , o algoritmo fornece uma maneira de reajustar os pesos da rede a partir de um sinal de erro entre a saída da rede O_n e a saída esperada, e.g. $Erro = (O_e - O_n)^2$. *Back-propagation* basicamente consiste em computar o gradiente do erro em função dos pesos das conexões dos neurônios. Utilizando a regra da cadeia, define-se a “contribuição” de cada peso para o valor de erro atual. O processo de otimização consiste em alterar os pesos gradualmente na direção oposta à do gradiente, de forma iterativa. O tamanho do passo em cada iteração é chamado de passo de aprendizagem (*learning rate*). Assim, o treinamento estimula o modelo a aprender quais características do vetor de entrada são relevantes para a classificação correta, para prever a saída desejada. Se a entrada são os valores de intensidade dos *pixels* de uma imagem, o modelo aprende a extrair *features* relevantes diretamente, sem a necessidade de um processo de extração prévio. Isso é o chamado de aprendizado *end-to-end*.

1.3. Metodologia

Nesta seção serão detalhados o conjunto de dados e as especificidades do treinamento de cada método descrito anteriormente.

1.3.1. Conjuntos de Dados – *Datasets*

Para medir o desempenho e comparar os diferentes métodos propostos, utilizamos dois conjuntos de dados distintos com fotos de frutas e vegetais. Ademais, o método de BoF necessita de algumas imagens para a construção do dicionário visual. Por isso, os dois conjuntos de dados são divididos em um conjunto para construção do dicionário BoF com 40% dos dados e o restante para classificação. Os métodos de HOG e RNs são treinados somente na divisão de classificação. Para todos os métodos, na divisão de classificação, os resultados são apresentados após 10 rodadas de validação cruzada com 20% dos dados reservados para validação, e o restante para treino.

1.3.1.1. *Dataset modificado da Unicamp*

O *dataset* da Unicamp [Rocha et al. 2010] apresenta imagens em um ambiente controlado, com fotos de frutas e vegetais em cima de uma mesa, com variações de iluminação, número e posição dos objetos. Originalmente continha 15 categorias distintas, porém, removemos a categoria *onion* (cebola) por julgarmos conter um número de imagens insuficientes para a aplicação do método. Também foram removidas algumas imagens

que continham mãos humanas e imagens que não continham objeto nenhum. Apesar dessas modificações, julgamos que o *dataset* permanece representativo e relevante para o cenário proposto. O *dataset* final contém um total de 2517 imagens nas seguintes categorias: *agata_potato* (199), *asterix_potato* (180), *cashew* (206), *diamond_peach* (211), *fuji_apple* (212), *granny_smith_apple* (154), *honeymelon* (135), *kiwi* (171), *nectarine* (246), *orange* (103), *plum* (257), *spanish_pear* (147); *taiti_lime* (105); *watermelon* (191). Desse total de imagens, 1015 foram utilizadas para construção do dicionário do BoF e 1502 foram utilizadas para classificação. A Figura 1.5 mostra alguns exemplos de imagens deste *dataset*.



Figure 1.5: Exemplos de imagens do *dataset* da Unicamp

1.3.1.2. Dataset Próprio

Para testar o desempenho dos métodos em um cenário mais desafiador, construímos um segundo *dataset*, onde as imagens continham uma presença maior de *background clutter*. Nossa *dataset* tem um total de 4 categorias, sendo 3 categorias de frutas (abacaxi, melão e banana) e uma categoria de *background* com imagens que não contém nenhuma fruta. Utilizamos uma câmera Sony DSC-H100 para fotografar as frutas no ambiente de nosso laboratório. O *dataset* contém um total de 351 imagens distribuídas nas seguintes categorias: *background* (89), banana (130), melão (56) e abacaxi (89). Desse total de imagens, 212 foram utilizadas para construção do dicionário do BoF e 139 para classificação. A categoria *background* contém imagens que não possuem nenhuma fruta. Nas 212 imagens para a construção do dicionário aplicamos um processo adicional, de remoção do fundo da imagem, de forma a restar apenas a própria fruta na imagem, garantindo que *features* extraídas realmente façam parte da textura das frutas. A Figura 1.6 apresenta alguns exemplos de imagens utilizadas para construção do dicionário visual do BoF, e a Figura 1.7 imagens utilizadas para classificação.



Figure 1.6: Imagens de nosso *dataset* usadas para construção do dicionário do BoF.

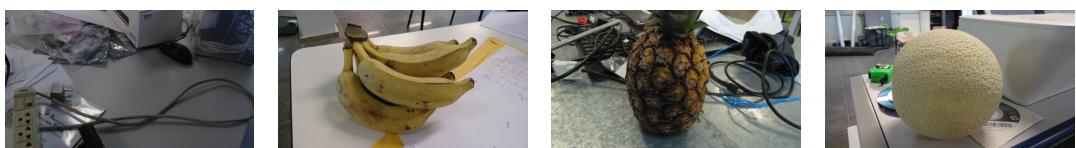


Figure 1.7: Imagens de nosso *dataset* utilizadas para classificação.

1.3.2. Construção do dicionário visual do BoF

Utilizando o conjunto de dados separados para construção do dicionário visual, para cada uma das categorias são computados os descritores (SIFT ou ORB) de todas as imagens. Para fins de manter um equilíbrio do número de *features* entre diferentes categorias, são utilizados o mesmo número de descritores de cada categoria. Isso é realizado ordenando os descritores em ordem decrescente de acordo com o parâmetro *response*, que representa a qualidade de um ponto de interesse, a probabilidade que ele possa ser reencontrado em imagens diferentes da mesma cena. A quantidade de descritores retirados de cada categoria é definida como 80% da categoria com o menor número de descritores. Utilizando extração de *features* ORB, a categoria de menor número para o *dataset* da Unicamp foi *spanish_pear* com 7474 e para o nosso *dataset* foi *melon* com 11654. Assim, o dicionário visual foi construído retirando, de cada categoria, 5979 *features* para o *dataset* da Unicamp e 9323 *features* para nosso próprio *dataset*. Para o método de extração com SIFT, a categoria de menor número também foi *spanish_pear* com 3006 *features* no *dataset* da Unicamp e *banana* com 26554 *features* em nosso próprio *dataset*. Da mesma maneira, foram retiradas, de cada categoria, 2404 *features* para o *dataset* da Unicamp e 21243 *features* para nosso próprio *dataset*. Para cada tipo de descritor, ORB ou SIFT, e para cada *dataset* foram construídos 13 modelos de *bag of features*, agrupando os dados com *K-Means* nos seguintes números de grupos: 50, 75, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900 e 1000.

1.3.3. Extração de descritores HOG

A extração dos descritores HOG é feita de maneira simplística, utilizando toda a imagem. Como previamente discutido na Sub-seção 1.2.2, o algoritmo possui uma série de parâmetros que definem o tamanho do vetor final. Neste trabalho, redimensionamos as imagens para resolução 128x128 e usamos blocos de tamanho 16x16 e células 8x8. O histograma de uma célula contém 9 *bins*. Cada bloco gera 4 histogramas com 9 *bins*, normalizados e concatenados um vetor de tamanho 36. Com um total de 64 blocos o comprimento final do descritor HOG é 2304. Esse vetor é utilizado diretamente pelo classificador SVM.

1.3.4. Classificador SVM

O classificador SVM é treinado utilizando o conjunto de dados separado para classificação com 10 rodadas de validação cruzada. Para cada modelo de BoF e para os descritores HOG foi realizada uma busca em grade pelos melhores parâmetros. Além disso, devido ao fato que o classificador SVM é sensível ao equilíbrio dos dados (número de exemplos por classe), para cada amostra do conjunto de treino é atribuído um peso de acordo com o número de exemplos de sua classe.

1.3.5. Modelos de CNN

Exploramos duas arquiteturas diferentes de CNNs profundas modernas. A primeira, conhecida como *AlexNet*, foi o primeiro modelo profundo que provou a eficiência dos mesmos, avançando o estado da arte significativamente [Krizhevsky et al. 2012]. *AlexNet* consiste em 5 camadas de CNN seguida por 2 camadas *feedforward*, com um total aproximadamente 60 milhões de parâmetros. Essa arquitetura foi treinada por 200 épocas e

o algoritmo otimizador utilizado foi *Stochastic Gradient Descent* (SGD). O tamanho de *batch* de 128 amostras e *learning rate* de 0.001 para o *dataset* da Unicamp e *batch* de 16 amostras e *learning rate* de 0.0001 para o *dataset* de nossa própria criação.

A segunda arquitetura que utilizamos é baseada na arquitetura *Tiny YOLO-V3* [Redmon et al. 2016] e possui 7 camadas de convolução seguida de uma camada *feedforward*, além de avanços recentes como camadas *batch normalization* [Ioffe and Szegedy 2015] após cada camada de convolução. Essa arquitetura possui aproximadamente 6 milhões de parâmetros, porém, é mais profunda, com 8 camadas escondidas. Utilizamos o mesmo otimizador, SGD, para essa arquitetura. Para o *dataset* da Unicamp treinamos por 30 épocas com *batch* de 32 amostras e *learning rate* 0.001. Para o *dataset* de nossa própria criação treinamos por 200 épocas com *batch* de 8 amostras e *learning rate* de 0.0001. O motivo da redução do tamanho do *batch* para nosso próprio *dataset* é devido a seu menor número de amostras.

1.4. Resultados

Esta seção apresenta a exatidão de cada método proposto, no conjunto de dados de validação, após 10 rodadas de validação cruzada. A Tabela 1.1 apresenta os resultados para o *dataset* da Unicamp e a Tabela 1.2.

Método	Exatidão
a – BoF-ORB - SVM	89.60 %
b – BoF-SIFT - SVM	93.30 %
c – HOG - SVM	96.68 %
d – AlexNet	90.88 %
e – Tiny YOLO-V3	97.69 %

Table 1.1: Exatidão de cada método para o *dataset* da Unicamp.

Método	Exatidão
a – BoF-ORB - SVM	82.22 %
b – BoF-SIFT - SVM	90.00 %
c – HOG - SVM	82.59 %
d – AlexNet	92.96 %
e – Tiny YOLO-V3	91.11 %

Table 1.2: Exatidão de cada método para o *dataset* de nossa própria criação.

1.5. Conclusão

Os resultados demonstram como a escolha do algoritmo de extração de *features* pode influenciar, mesmo utilizando o mesmo tipo de classificador. Comparando os métodos na Tabela 1.1-a com Tabela 1.1-b e Tabela 1.2-a com Tabela 1.2-b, pode-se observar claramente que, mesmo utilizando o mesmo modelo de representação BoF, a exatidão das *features* SIFT é significativamente melhor que *features* ORB em ambos *datasets*. Curiosamente, o método HOG sofreu uma grande diferença de desempenho entre os dois

datasets como observado pela Tabela 1.1-c e Tabela 1.2-c, sugerindo que o método não é robusto à imagens com presença significativa de *background*. Finalmente, os métodos baseados em RNs atingiram o melhor desempenho em ambos os *datasets*, *Tiny YOLO-V3* para o *dataset* da Unicamp na Tabela 1.1-e, e *AlexNet* no *dataset* de nossa própria criação na Tabela 1.2-d.

Este trabalho mostra que os modelos tradicionais com extração de *features* manuais podem atingir um desempenho comparável a modelos baseados em RNs. Com um pequeno número de parâmetros, sem a necessidade de uma GPU para treinamento e uma inferência rápida, os modelos tradicionais podem ser uma escolha superior para determinados cenários e configurações de *hardware*, como sistemas embarcados. Todavia, é necessário um processo prévio de extração de *features*, cuja escolha e implementação pode ter um impacto no desempenho final. Em contrapartida, considerando o estado e popularidade atual de *frameworks* voltados para o desenvolvimento de modelos de RNs há uma considerável facilidade no treinamento e aplicação prática de tais modelos, ainda que mais complexos teoricamente e em número de parâmetros. Outra vantagem importante para RNs é a possibilidade do aprendizado *end-to-end*, eliminando o processo de extração de *features* e operando diretamente sobre os dados originais.

1.6. Agradecimentos

Este trabalho foi financeiramente apoiado pelo “Chinese Language and Technology Center” da National Taiwan Normal University (NTNU) com recursos do Ministério da Educação de Taiwan e Ministério de Ciência e Tecnologia de Taiwan sob os números MOST 108-2634-F-003-002, MOST 108-2634-F-003-003, MOST 108-2634-F-003-004 e MOST 107-2811-E-003-503. Somos gratos ao Centro Nacional de Computação de Alto Desempenho de Taiwan pelo seus recursos computacionais. Agradecemos também a NVIDIA Corporation pela doação de uma GPU Titan Xp, utilizada nesta pesquisa.

References

- [Akbari Fard et al. 2016] Akbari Fard, M., Hadadi, H., and Tavakoli Targhi, A. (2016). Fruits and vegetables calorie counter using convolutional neural networks. In *Proceedings of the 6th International Conference on Digital Health Conference*, pages 121–122. ACM.
- [Anthimopoulos et al. 2014] Anthimopoulos, M., Gianola, L., Scarnato, L., Diem, P., and Mougiaikakou, S. G. (2014). A food recognition system for diabetic patients based on an optimized bag-of-features model. *IEEE J. Biomedical and Health Informatics*, 18(4):1261–1271.
- [Bishop et al. 1995] Bishop, C. M. et al. (1995). *Neural networks for pattern recognition*. Oxford university press.
- [Burges 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167.

- [Dalal and Triggs 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.
- [Duda et al. 2012] Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.
- [Ioffe and Szegedy 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- [Jean et al. 2016] Jean, N., Burke, M., Xie, M., Davis, W. M., Lobell, D. B., and Ermon, S. (2016). Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794.
- [Ke and Sukthankar 2004] Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE.
- [Kourou et al. 2015] Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17.
- [Krizhevsky et al. 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [LeCun et al. 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.
- [Lee et al. 2015] Lee, J.-C., Chang, C.-P., and Chen, W.-K. (2015). Detection of copy-move image forgery using histogram of orientated gradients. *Information Sciences*, 321:250–262.
- [Libbrecht and Noble 2015] Libbrecht, M. W. and Noble, W. S. (2015). Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321.
- [Lowe 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.
- [Minaee et al. 2017] Minaee, S., Wang, S., Wang, Y., Chung, S., Wang, X., Fieremans, E., Flanagan, S., Rath, J., and Lui, Y. W. (2017). Identifying mild traumatic brain injury patients from mr images using bag of visual words. In *2017 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–5. IEEE.
- [Mohri et al. 2018] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

- [Perez and Wang 2017] Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.
- [Redmon et al. 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.
- [Rocha et al. 2010] Rocha, A., Hauagge, D. C., Wainer, J., and Goldenstein, S. (2010). Automatic fruit and vegetable classification from images. *Computers and Electronics in Agriculture*, 70(1):96–104.
- [Rublee et al. 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. R. (2011). ORB: An efficient alternative to SIFT or SURF. In *ICCV*, volume 11, page 2. Citeseer.
- [Rumelhart et al. 1988] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.
- [Sebastiani 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.
- [Tian et al. 2016] Tian, S., Bhattacharya, U., Lu, S., Su, B., Wang, Q., Wei, X., Lu, Y., and Tan, C. L. (2016). Multilingual scene character recognition with co-occurrence of histogram of oriented gradients. *Pattern Recognition*, 51:125–134.
- [Zhang et al. 2014] Zhang, Y., Wang, S., Ji, G., and Phillips, P. (2014). Fruit classification using computer vision and feedforward neural network. *Journal of Food Engineering*, 143:167 – 177.
- [Zhang et al. 2017] Zhang, Y.-D., Dong, Z., Chen, X., Jia, W., Du, S., Muhammad, K., and Wang, S.-H. (2017). Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. *Multimedia Tools and Applications*, pages 1–20.