Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Antonio Augusto Abello

Uso de Redes Neurais Mistas Para Classificação de Plâncton

São Paulo
Dezembro 2017

Uso de Redes Neurais Mistas Para Classificação de Planktôn

Monografia final do curso
MAC0499 - Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Roberto Hirata Jr.

São Paulo
Dezembro 2017

Agradecimentos

Agradeço à minha família pelo suporte de todos os tipos durante a graduação e especialmente durante a produção deste trabalho.

A meu orientador Roberto Hirata Jr., por ter sido e continuado a ser um verdadeiro professor, instigar minha curiosidade e buscar extrair o melhor de mim, dentro e fora da sala de aula.

Adicionalmente à profª Nina S.T. Hirata, profº Rubens M Lopes, e os colegas Francisco Caio Maia Rodrigues e Leandro T. De La Cruz pela oportunidade de colaborar em pesquisa e companheirismo.

E finalmente, agradeço a todos meus colegas do IME-USP que me acompanharam nestes anos pela maior lição de todas: não é necessário, nem possível, passar por isto sozinho. Juntos fortalecemos uns aos outros.

Abstract

Convolutional Neural Networks (CNNs) have become the state of the art in a number of computer vision task, specially in supervised classification. This is mainly due to them being domain-agnostic, working well for any type of image given there are sufficient examples.

Nevertheless, they can incur in information loss, wheter by technical reasons (the necessity of a fixed image size) or by it's own indifference to the domain. In various cases there is expert knowledge, used to built traditional classifiers based on feature engineering, that remains underused in Deep Learning approaches. There is, then, the possibility of reinserting this knowledge in modern classifiers.

In this work we propose a mixed model of CNN and classifier by feature engineering. We develop a way of training it efficiently and achieve results that surpasses the individual performance of both approaches.

We run experiments on a plankton dataset, taking advantage of expert knowledge already existent and compiled in various works of an ongoing collaboration between IME (Mathematics and Statistics Institute) and LAPS-IO (Laboratory of Plankton Systems - Oceanographic Institute).

Keywords: Computer Vision, Deep Learning, Plankton Classification.

Resumo

Redes Neurais Convolucionais (CNNs) se tornaram o estado da arte em diversas tarefas de visão computacional, em específico a classificação supervisionada. Isto se deve principalmente por serem agnósticas de domínio, funcionando bem para qualquer tipo de imagem desde que haja exemplos o suficiente.

No entanto, elas podem incorrer em perda de informação, seja por motivos técnicos (a necessidade de um tamanho padronizado de imagens), seja pela própria indiferença ao domínio. Em diversos casos há conhecimento de especialistas, usado para construir classificadores tradicionais baseados em extração de características, que fica subaproveitado em abordagens de Deep Learning. Surge então a possibilidade de se reinserir esse connhecimento nos classificadores atuais.

Neste trabalho propomos um modelo misto de CNN e classificador por extração de características. Descobrimos uma forma de treiná-lo e conseguimos resultados que superam a performance individual de ambas as abordagens.

Realizamos experimentos sobre um dataset de plâncton, aproveitando-nos do conhecimento de especialista já existente e compilado em diversos trabalhos de uma colaboração já existente entre o IME e o LAPS-IO (Laboratório de Estudos Planctônicos do Instituto Oceanográfico).

Palavras-chave: Visão Computacional, Deep Learning, Classificação de Plâncton.

Conteúdo

Lista de Abreviações

LAPS	Laboratory of Plankton Systems
IME	Instituto de Matemática e Estatística
GPGPU	General Purpose Graphics Processing Unit
CNN	Convolutional Neural Network
FC	Fully Connected (Layer)
ReLU	Rectified Linear Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge

Capítulo 1

Introdução

Tradicionalmente, a abordagem de classificação supervisionada de imagens se baseia em extratores de características: uma série de estatísticas e informações relevantes para a classificação são extraídas da imagem e a representam como um vetor de alta dimensão para os algoritmos de classificação.

Por mais que grande parte dos extratores possam ser aplicados a diversos domínios sem problemas, para alcançar uma performance satisfatória é necessário a seleção e por vezes até a criação de novos extratores pensando nas características particulares do domínio. A expansão de técnicas de Visão Computacional e Machine Learning a novos horizontes estava, então, atrelada ao conhecimento disponível de especialistas de cada novo domínio.

Diferentemente desta abordagem, o conjunto de técnicas chamadas de Deep Learning torna a extração de características parte do algoritmo de aprendizado. No caso de imagens, Redes Neurais Convolucionais[20] (CNNs) processam imagens completas, aprendendo durante o treino um espaço de representação útil para classificar exemplos. Em contrapartida, estes modelos precisam de bastante poder de processamento, em geral de placas de vídeo e GPGPU, e um número elevado de exemplos classificados para obter bons resultados.

Desde 2012, quando um modelo de Rede Neural Convolucional ganhou por grande margem a ILSVRC, a maior competição anual de classificação supervisionada e detecção de objetos, as abordagens chamadas de Deep Learning tomaram de assalto o campo de visão computacional[19]. Essas abordagens se popularizaram em seguida por sua alta versatilidade e aplicabilidade em diversos contextos, dado que houvessem dados o suficiente para garantir um bom aprendizado do espaço de representação.

Nem todos os contextos possuem uma disponibilidade para produzir uma grande quantidade de exemplos classificados, no entanto. Há às vezes dificuldade tanto de aquisição e produção de exemplos como de disponibilidade de trabalho humano para classificá-los. Além disso, em domínios com uma história já existente de pesquisa em classificação, o conhecimento de especialista produzido na escolha e popularização de determinados extratores de característica fica subutilizado (ou não utilizado). CNNs também podem incorrer em perda de informação por razões técnicas, como a necessidade de um tamanho padrão de imagem.

É possível, então, pensar em uma forma de reintegrar esse conhecimento aos modelos profundos para produzir melhores classificadores e cobrir eventuais perdas de informação. Por outro lado, podemos pensar em como usar os modelos profundos para melhorar classificadores tradicionais já existentes, onde não há exemplos o suficiente para garantir uma boa performance de CNNs sozinhas.

Para isso, estudamos neste trabalho Redes Neurais Mistas, que são redes neurais convolucionais que recebem imagens e vetores de características como entrada. Buscamos estudar como o conhecimento humano interage com o conhecimento extraído pelas CNNs, quão re-

dundantes ou ortogonais são, e como certificar que o treinamento consiga utilizar as duas fontes de conhecimento da melhor forma.

O problema em que aplicamos esses modelos é a classificação supervisionada de plâncton. Nomeia-se plâncton um conjunto de micro-organismos de água doce ou salgada que constitui a base da cadeia alimentar nos seus respectivos ecossistemas. Além disso, esses organismos tem papel crucial no ciclo de carbono e são os maiores produtores de oxigênio no planeta[17]. A avaliação automática da presença, diversidade, e saúde de organismos planctônicos em diversos corpos de água é, portanto, de grande importância e interesse para cientistas.

A questão da classificação supervisionada de plâncton é também um dos domínios com grande histórico de estudo anterior [6][16][34]. Em especial, há uma colaboração em andamento do Laboratório eScience, no IME-USP, com o Laboratório de Estudos de Sistemas Planctônicos, do Instituto Oceanográfico (LAPS-IO) que gerou, entre outras coisas, dois trabalhos de mestrado[14] [22]. Tomamos, portanto, a oportunidade de começar nosso trabalho não do zero, mas a partir do rico trabalho já disponível.

Capítulo 2

Revisão Bibliográfica

O problema de classificação automática de plâncton já é relativamente bem estudado. Em geral ele não é tratado separadamente do workflow de aquisição e tratamento das imagens, ou até mesmo do hardware utilizado. Isto acontece porque, por vezes, o sistema de aquisição já tem métodos de classificação ou extração de características integrados como em [16][6][34][12]. Em outros casos, como em [22], a classificação é parte de um sistema maior de imageamento e monitoramento de plâncton.

Há também, no entanto, artigos focados somente na questão de classificação de plâncton e suas peculiaridades, como o efeito da escolha do algoritmo de segmentação[14], o número de classes consideradas[13], e certas simetrias estruturais do espaço de exemplos [9]. Em [12] se faz uma pesquisa sobre o número de exemplos necessários para atingir bons resultados, uma comparação entre a performance de diversos algoritmos disponíveis e chega até a citar Redes Neurais Convolucionais, mas avalia não ser necessário dado a quantidade de parâmetros, e necessidade de tempo de processamento.

Este trabalho é em grande parte uma continuação dos esforços em uma colaboração corrente do laboratório de e-Science do IME-USP com o Laboratório de Estudo de Sistemas Planctônicos (LAPS), que já produziu duas dissertações de mestrado. De [22], temos um estudo sistemático sobre classificação de plâncton, incluindo uma lista de características extraídas e estudos sobre o efeito de diversas formas de seleção que vão aparecer novamente em [14] e que vão ser a base para as características utilizadas neste trabalho. De [14], temos um estudo sobre diferentes algoritmos de segmentação, sua implementação e seu efeito geral sobre os algoritmos de classificação. Vamos seguir suas conclusões para escolher, implementar e testar diversos algoritmos de segmentação.

Deep Learning é uma área de pesquisa bem recente. O primeiro grande livro sistematizando e delimitando a área é do final de 2016, publicado em 2017 [15], mas aplicações e técnicas hoje consideradas parte da área já existem há bastante tempo. As bases teóricas para Redes Neurais Convolucionais existem desde 1998[20][21]. Mas, sem nem o poder de processamento, nem o tamanho dos datasets, este modelo demorou a ser aplicado em larga escala fora de sua aplicação inicial de reconhecimento de dígitos[20].

A partir de 2012 CNNs ganharam força quando um modelo profundo, denominado AlexNet, ganhou com larga margem a tradicional competição de classificação e detecção de objetos ImageNet[19]. Este modelo também introduziu várias novidades que se tornaram padrão nas arquiteturas de redes neurais convolucionais, como a escolha de Rectified Linear Units (ReLU) para função de ativação, o uso de Dropout para regularização, e normalização entre camadas. Depois deste ano, a competição foi dominada por modelos profundos, sendo o maior foco subsequente o aumento da profundidade das redes e o design de novos tipos de arquitetura[30][28].

Ainda assim, a adoção de técnicas de Deep Learning ao problema de classificação de plâncton foi demorada. Um ponto importante de inflexão foi o "National Data Science Bowl", competição de classificação supervisionada de plâncton realizada pelo Kaggle que, em conjunto com a Universidade de Oregon, disponibilizou um grande dataset [7]. A competição foi bem produtiva em gerar publicações subsequentes de equipes vencedoras. O artigo [9] descreve inovações em arquitetura que permitiram construir classificadores robustos a rotação, algo especialmente importante para classificação de plâncton. Já [26] descreve uma técnica que permite decidir hiperparâmetros a partir da otimização de critérios de capacidade de aprendizado e desenvolve uma arquitetura capaz de usar diferentes tamanhos de entrada de imagem, para mitigar a perda de informação na padronização de tamanho de imagens.

Dado o sucesso da aplicação de Deep Learning para a classificação de plâncton nesta competição, naturalmente começou-se a se perguntar como trazer as vantagens destes métodos para datasets locais, com menos exemplos e outras dificuldades próprias. Motivado pelo sucesso da abordagem de Transfer Learning, tanto antes de Deep Learning [24] como no próprio contexto de CNNs [4][32], pesquisadores conseguiram com sucesso usar o dataset disponibilizado pelo Kaggle para ajudar a treinar classificadores para outros datasets. Nesse contexto, [23] mostrou a viabilidade dessa técnica, enquanto que uma publicação co-autorada pelo autor deste trabalho aprofundou os estudos sobre ela em situações reais[27].

Uma outra ideia para usar CNNs em datasets menores é tentar adaptar o modelo para fazê-lo funcionar nestas condições. [8], por exemplo, desenvolve uma arquitetura, ZooplanktoNet, que consegue resultados satisfatórios em um dataset de cerca de 9000 exemplos, pequeno comparado ao dataset do Kaggle, de cerca de 30.000. Outra abordagem possível, explorada neste trabalho, é misturar as CNNs com classificadores tradicionais, baseados em extração de características.

A ideia de combinar features específicas de domínio com Deep Learning já foi aplicada com sucesso em diversos contextos, como detecção de mitose em câncer de mama[31] e reconhecimento de expressões faciais[2]. Um estudo comparando features projetadas por humanos e aprendidas automaticamente para reconhecimento de gênero de pedestres concluiu também que CNNs podem aprender features úteis mesmo se forem pequenas e aplicadas a datasets pequenos[3], mostrando a viabilidade deste trabalho.

Não há, até onde se encontrou, pesquisas sobre CNNs mistas em termos mais abstratos e gerais, nem sobre a combinação de features manuais com aprendidas no problema de classificação de plâncton em si. No entanto, a equipe vencedora do National Data Science Bowl reportou utilizar features manuais em seu pipeline de classificação[1], relatando uma surpreendente ortogonalidade entre os dois métodos.

[1] ver: http://benanne.github.io/2015/03/17/plankton.html

Capítulo 3

Fundamentos Teóricos

3.1 O problema da classificação

O problema da classificação supervisionada consiste em, dado X um espaço possível de dados de entrada e Y um espaço discreto de classes, encontrar a melhor aproximação possível para a função ideal $f : X \to Y$ que corretamente atribui a cada exemplo de X sua classe verdadeira. Essa função é aproximada a partir de um dataset D de exemplos (x_i, y_i) previamente classificados. Por vezes também se usa uma interpretação probabilística de f como a probabilidade condicional $P(Y|X)$, o que permite pensar a abordagem de aprendizado como estimação por máxima verossimilhança.

3.2 Generalização e Overfitting

O objetivo, no entanto, não é garantir uma boa performance do modelo dentro do dataset de treino, mas sim em exemplos não vistos durante o treinamento (o que se chama de generalização). A teoria de aprendizagem estatística estuda as condições necessárias para que isso aconteça. Importantes resultados teóricos como a dimensão VC e o Bias-Variance Trade-off[1] correlacionam a capacidade de generalização com a capacidade de representação do modelo e a quantidade de dados disponível.

Aparece aí um trade-off: modelos mais complexos podem aproximar f melhor, mas também podem se sobre-especializar ao dataset, eventualmente aprendendo o ruído presente nos dados, o que se chama de overfitting. Por essa mesma razão, esses modelos são mais sensíveis a mudanças no conjunto de treino e podem apresentar uma variância maior para diferentes divisões de dataset ou até entre modelos treinados no mesmo dataset. Por outro lado, modelos simples demais podem se mostrar incapazes de se aproximar de f, possuindo um viés sistemático.

A principal ferramenta para diagnosticar a capacidade de generalização é a divisão do dataset em conjuntos de treino e de teste. O modelo é treinado somente no conjunto de treino e a acurácia no conjunto de teste é tomada como boa estimativa da acurácia em outros dados. Quando é necessário otimizar a configuração dos modelos (hiperparâmetros) é comum ainda dividir o dataset de treino entre treino e validação, de modo a otimizar os hiperparâmetros testando seu efeito sobre a performance no conjunto de validação e usando o conjunto de teste somente para estimar a generalização.

Há ainda diversas ferramentas para prevenir o overfitting e melhorar a generalização. Chamam-se medidas de regularização qualquer medida com objetivo de melhorar a performance do modelo exclusivamente no conjunto de teste, muitas vezes às custas do erro dentro

do dataset de treino. Exemplos de estratégias de regularização que podem ser aplicadas a virtualmente qualquer modelo:

- **Penalizações de funções de erro** - inclusão de um fator baseado na norma ou número de parâmetros do modelo à função de custo a ser otimizada
- **Early Stopping** - acompanhar o desempenho do modelo no conjunto de treino ou validação e parar o treino caso não apresente mais melhora ou apresente piora

Há também estratégias de regularização específicas para certos tipos de modelo, que serão descritas mais à frente.

3.3 Modelos e Algoritmos

Em geral, se limita a busca entre possíveis funções modeladoras a alguma família de funções, o que define um algoritmo de aprendizado de máquina. Neste estudo trabalharemos com duas famílias de modelos: regressão logística e redes neurais.

3.3.1 Regressão Logística Multinomial

Regressão logística é um modelo originalmente estatístico que pode ser usado também em tarefas de aprendizado de máquina. É um modelo que explicitamente tenta aproximar a probabilidade $P(Y|X)$ como um produto interno entre as variáveis de entrada e um vetor de pesos passados pela função logística, que garante que a saída seja de fato uma probabilidade.

A versão multinomial da regressão logística estende o modelo para problemas com múltiplas classes. Ao invés de um vetor de pesos se usa uma matriz de modo que a saída tenha a mesma dimensão que o número de classes. Ao invés da função logística se usa a função *softmax*, uma generalização da função logística que torna a saída um vetor de probabilidades por classe

$$\sigma(z)_i = \frac{e^{z_i}}{\sum e^{z_j}} \tag{3.1}$$

Figura 3.1: *Função Softmax*

$$f(x; W) = \sigma(Wx + b) \tag{3.2}$$

Figura 3.2: *Regressão Logística parametrizada*

3.3.2 Redes Neurais

Redes Neurais podem ser entendidas como uma sucessão de regressões logísticas, com algumas diferenças. Ao invés de usar a função softmax, usa-se outra função não-linear para as camadas intermediárias, também chamada de função de ativação. O objetivo ao adicionar diversas camadas é aumentar o poder de expressividade do modelo, controlado pelo número de camadas e o tamanho das camadas intermediárias.

$$f(x;\theta) = \sigma(W_2\phi(W_1x + b_1) + b_2) \tag{3.3}$$

Figura 3.3: *Rede Neural com uma camada intermediária. σ é a função softmax já definida e ϕ é uma função não-linear. Candidatos comuns para ϕ são a função sigmóide, tangente hiperbólica e ReLU ($max(0, x)$)*

3.3.3 Função de Erro

Para quantificar o ajuste do modelo, se define uma função de erro em função da previsão dada. Em geral se define primeiro a função de erro para um exemplo, e depois se generaliza a fórmula para um grupo de previsões. Neste trabalho, aproveitando-nos da interpretação probabilística do problema de classificação, se utiliza a entropia cruzada como medida de erro dos modelos. Esta medida intuitivamente mede a distância entre duas distribuições, no caso, a distribuição real e a distribuição prevista pelo algoritmo (ver [5], p. 206 para uma explicação mais detalhada).

$$L(y, \hat{y}) = -\sum_k y_k \log(\hat{y}_k) \tag{3.4}$$

Figura 3.4: *Entropia Cruzada*

Onde y é a distribuição de probabilidades real (ou seja, um vetor com 1 de probabilidade para a classe correta e 0 para as demais) e $\hat{y}$ é a previsão do classificador. Em específico, $\hat{y}_k = P(y = k|x;\theta)$. Desta forma podemos simplificar a função acima como

$$L(y, \hat{y}) = -\log(P(y = k|x;\theta)) \tag{3.5}$$

Figura 3.5: *Entropia Cruzada (simplificada)*

Note-se que o erro é nulo quando a probabilidade prevista para a classe correta é 1 (ou seja, quando o classificador acerta) e tende a menos infinito quando ela vai a 0. Há um problema numérico se o classificador de fato dá probabilidade 0 para a classe correta, uma vez que a função logaritmo não está definida neste ponto. Por isso, em geral, se adiciona um ϵ a esta probabilidade durante o cálculo do erro. Para um conjunto de dados se define o erro do classificador simplesmente como a média simples dos erros individuais.

3.3.4 Otimização

Uma vez definida a função de erro, o processo de ajustar o modelo aos dados pode ser definido como um problema de otimização, minimizar a função de erro com relação aos parâmetros. Para isso podemos empregar métodos de otimização não-linear e convexa. Entre os métodos mais utilizados no contexto de Machine Learning estão o método do gradiente e o método de Newton, que aproximam a função de seu mínimo iterativamente, ao mudar os parâmetros do modelo seguindo uma regra fixa, chamada de regra de atualização.

A regra de atualização do método do gradiente se justifica pelo uso de noções bem intuitivas de cálculo: o gradiente de uma função em um ponto determinado é a direção de maior crescimento local. Logo, uma boa direção para minimizar o erro é o contrário do gradiente, multiplicado por uma constante definida como a "taxa de aprendizado"(learning rate)

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L(y, \hat{y}) \tag{3.6}$$

Figura 3.6: *Regra de atualização do método do gradiente para um parâmetro θ*

O método de Newton usa aproximações de Taylor de segundo grau ao redor de um ponto para encontrar a direção ótima. A regra de atualização é a que leva a função ao mínimo global único desta aproximação dispensando, portanto, o uso de uma taxa de aprendizado.

$$\theta_{t+1} = \theta_t - \{\nabla^2_\theta L(y, \hat{y})\}^{-1} \nabla_\theta L(y, \hat{y}) \tag{3.7}$$

Figura 3.7: *Regra de atualização do método do Newton para um parâmetro θ*

O método de Newton leva geralmente a uma convergência mais rápida e melhor (em casos que a função de erro não tenha um único mínimo). No entanto, a necessidade de calcular o Hessiano e invertê-lo pode trazer um grande custo em processamento e memória, em especial para modelos com muitos parâmetros. Assim, raramente se usa métodos de segunda ordem para redes neurais, preferindo-se o método do gradiente e variantes.

Um importante tipo de variante do método do gradiente são os chamados métodos adaptativos. Eles mantêm uma taxa de aprendizado para cada parâmetro e a aumentam ou diminuem de acordo com a taxa de mudança do parâmetro em si. Dessa forma, o algoritmo dispensa uma escolha muito minuciosa de taxas de aprendizado. O primeiro destes algoritmos, usado neste trabalho é o ADAGRAD[11].

Para determinar a atualização das redes neurais é necessário ainda o algoritmo de backpropagation[21], que usa de aplicações recursivas da regra da cadeia para calcular o gradiente de cada camada.

3.4 Engenharia de Features

Por restrições tanto de processamento como de complexidade de modelos, a abordagem tradicional de classificação de imagens não utiliza as imagens inteiras. Ao invés disso, existe um passo intermediário em que características consideradas informativas são extraídas das imagens. Cada imagem então é representada por um vetor de características em R^n antes de serem classificadas.

Exemplos de características usadas:

- Estatísticas do nível de cinza: média, desvio padrão, demais momentos centrais
- Histograma
- Momentos de imagem, momentos de Hu (construções invariantes à rotação a partir dos momentos de imagem)

- Descritores de Forma

Estes últimos são especialmente importantes para este trabalho. São diversas quantidades descrevendo aspectos da forma do objeto em questão, tais como área, circularidade, elongamento, entre outros. Esses números são na maior parte calculados usando as dimensões de formas ajustadas ao contorno do objeto, tais como: menor retângulo, menor elipse, menor círculo entre outros.

Para calcular esses descritores, então, é necessário isolar o contorno de um objeto do resto da imagem, processo que tem o nome de segmentação. A qualidade da segmentação é crucial para a corretude e informatividade dos descritores de forma, assunto estudado extensivamente no próprio contexto de plâncton.[14]

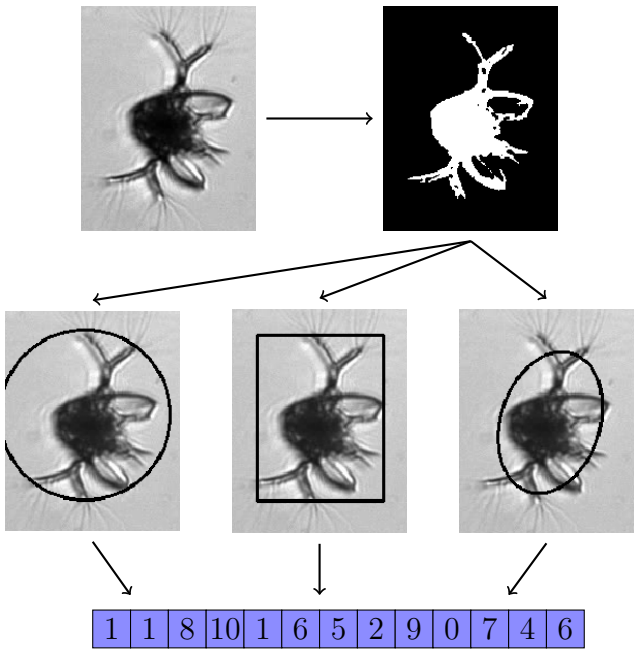

Figura 3.8: *Esquema demonstrando extração de descritores de forma*

3.4.1 Algoritmos de Segmentação

Neste trabalho, exploramos dois tipos de algoritmos de segmentação: os baseados em threshold automáticos e o baseado em Watershed com marcadores.

Os algoritmos de threshold automáticos buscam uma intensidade ótima tal que todo pixel com intensidade maior ou igual que ela pertence ao fundo e todo pixel com intensidade menor pertence ao objeto. O que diferencia os dois algoritmos utilizados é o critério de seleção: no caso do algoritmo de Otsu, busca-se o threshold que maximiza a variância entre as duas classes. No caso do algoritmo de Yen se maximiza um critério complexo de correlação.

O algoritmo de Watershed originalmente serve para descrever a topologia de uma imagem ao interpretar a intensidade de cinza como "altura"e simular uma "enchente"a partir dos mínimos regionais, sendo os pontos de encontro da "água"as bordas que definem a topologia na imagem. No entanto, como mostrado por [14], essa técnica pode ser usada para produzir uma segmentação da seguinte forma: se define da forma mais conservadora possível um threshold mínimo para pixels do objeto e um máximo para pixels do fundo, criando ainda um terceiro set de pixels ambíguos. Usando o algoritmo do Watershed a partir desses marcadores

ao invés dos mínimos regionais pode-se definir o contorno do objeto, deduzindo quais pixels ambíguos são fundo ou parte do objeto.

Abaixo mostramos uma comparação entre os algoritmos de segmentação para uma imagem de exemplo.

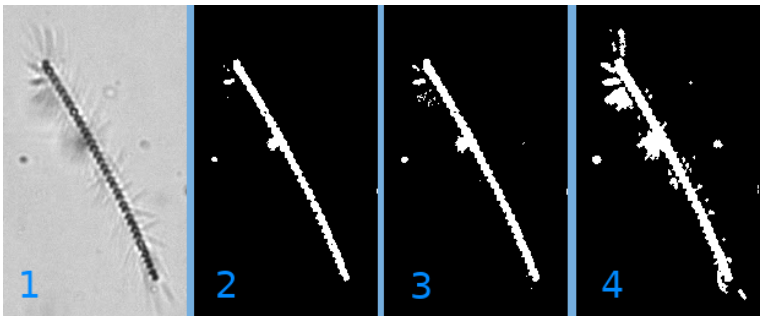

Figura 3.9: *Imagem exemplo (1) e segmentações de Otsu (2), Yen (3), e Watershed por marcadores (4)*

3.5 Deep Learning

Deep Learning é a abordagem de aprendizado de máquina que inclui o aprendizado de representações abstratas dos dados de entrada[15] como parte do treinamento. Desse modo, não se depende da construção de espaços de representação por humanos como no caso de engenharia de features. No processamento de imagens isso geralmente implica o uso de Redes Neurais Convolucionais, capazes de processar imagens inteiras.

3.5.1 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNNs) são uma extensão de Redes Neurais baseada principalmente na ideia de filtros convolucionais. Em visão computacional já se usa a operação de convolução de um filtro pequeno por uma imagem para extração de características, sendo o desenvolvimento de diversos filtros (ou kernels) objeto de grande interesse teórico.

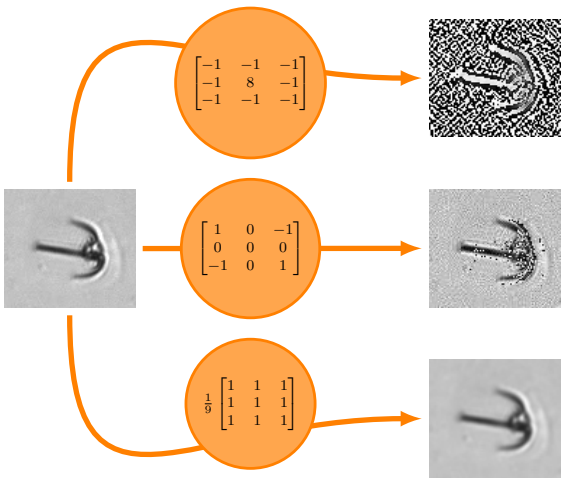

Figura 3.10: *Exemplo de aplicação de filtros em uma imagem*

Em CNNs se usam, então, diversos filtros ajustáveis, organizados em camadas, de forma análoga às matrizes ajustáveis de uma Rede Neural normal. Entre uma camada e outra há

também uma função não-linear de ativação, geralmente ReLU ou um variante.

Além disso, a CNN vai gradualmente diminuindo a dimensão do input por meio de convoluções com passos (stride) maiores ou por meio de camadas de pooling. A camada de pooling funciona similarmente à camada convolucional: há uma "janela"que é convoluída pela imagem aplicando uma função e computando uma saída. Há diversos tipos de pooling dependendo dessa função: max-pooling, average pooling, l2-pooling, entre outros.

Por fim, após progressivamente extrair informações relevantes da imagem (camadas de convolução) e diminuir a dimensão da imagem (camadas de pooling), a saída de uma camada intermediária é interpretada como vetor de características e então classificada por uma rede neural (camada denominada de Fully Connected, Densa, entre outros).

A rede é ajustada por meio de backpropagation e método do gradiente, da mesma forma que a rede neural comum e a regressão logística.

Há diversas formas de arquitetura de redes, e a pesquisa em CNN está em sua maior parte focada em desenvolver novos tipos de camadas e arquiteturas. No entanto, para este trabalho consideramos uma arquitetura mais básica, descrita por este algoritmo:

Algorithm 1 Construção de CNN deste trabalho (hiperparâmetros não mostrados)

```
1:  procedure BuildCNN(imsize)
2:      network ← InputLayer(imsize)
3:      inputsize ← imsize
4:      while inputsize > 8 do                    ▷ Diminui input até ≤ 8x8
5:          network ← ConvLayer(network)
6:          network ← ReLULayer(network)
7:          network ← MaxPoolLayer(network)
8:          inputsize ← inputsize/2
9:      network ← DropoutLayer(network)           ▷ Rede Neural normal
10:     network ← DenseLayer(network)
11:     network ← ReLULayer(network)
12:     network ← DropoutLayer(network)
13:     network ← DenseLayer(network)
14:     network ← SoftmaxLayer(network)
        return network
```

Onde "imsize"é o tamanho da imagem (supõe-se que ambas as dimensões são iguais), e as diversas funções terminadas por "Layer"são funções que adicionam uma nova camada à rede e devolvem uma representação da rede com a nova camada.

3.5.2 Deep Learning e Generalização

Modelos de Deep Learning são geralmente muito mais complexos e maiores que os modelos tradicionais de Machine Learning, exigindo então muitos mais dados para convergir adequadamente. Ainda assim, há dúvidas de como ou por que modelos profundos conseguem generalizar, dado que a capacidade representativa dos modelos e o sucesso que eles encontram não seguem os resultados teóricos estabelecidos da área. A conclusão corrente é de que para entender Deep Learning é necessário repensar como entendemos generalização[33].

Essa característica também aumenta bastante a importância de regularização. De fato, em geral não se busca diminuir o tamanho dos modelos para reduzir overfitting, mas sim aumentear e melhorar as técnicas de regularização usadas.

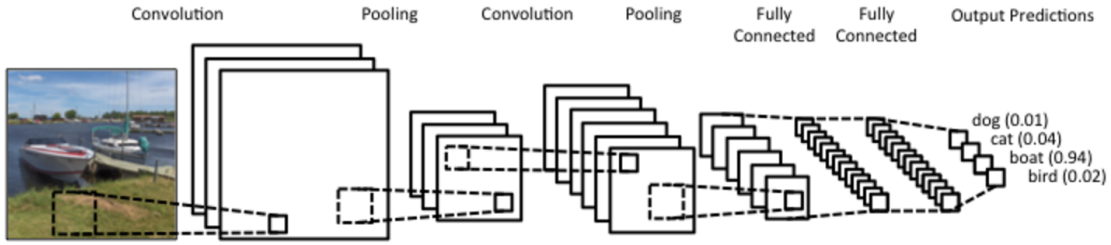

Figura 3.11: *Esquema de uma arquitetura de CNN, retirado de: http://www.wildml.com/2015/11/understanding- convolutional-neural-networks-for-nlp/*

Entre as técnicas de regularização criadas para esse propósito está o Dropout, feito para redes neurais profundas e convolucionais. A ideia é passar a saída de camadas por uma máscara aleatória que zera alguns componentes a cada iteração de treino, assim favorecendo soluções mais robustas a ruído e falta de informação. No paper em que se apresenta essa medida, argumenta-se que, na prática, o Dropout é equivalente a treinar uma enorme ensemble com as subredes induzidas pelas máscaras [29].

Outra técnica bastante utilizada em aplicações reais e neste trabalho é a aumentação de dados. Em especial quando se trabalha com imagens, mas não limitado a isto, se aplica transformações aos exemplos para produzir novos exemplos, aumentando o dataset na prática e tornando o classificador robusto à presença dessas transformações nos testes.

Capítulo 4

Proposta

4.1 Redes Neurais Mistas

O objeto de estudo principal deste trabalho são "Redes Neurais Mistas", um modelo que tenta juntar características do paradigma clássico de classificação de imagens com as vantagens recentes de Deep Learning. Essa rede tem dois canais de entrada: um para imagens, e outro para características, que são concatenadas à saída de uma camada intermediária.

Dessa maneira, por ter uma parte de CNN, o modelo pode processar as imagens inteiras, aprendendo a extrair características do dataset. Além disso, por processar também as features, o modelo pode contar com informação que é perdida no uso de CNNs puras, como o tamanho da imagem, e informação específica do domínio, proveniente do conhecimento teórico e prático que levou ao surgimento e popularização destas features particulares. Por outro lado, o modelo pode se aproveitar desse conhecimento de domínio como ponto de partida e (se espera) pode levar a uma melhora automática, sem necessidade de intervenção humana.

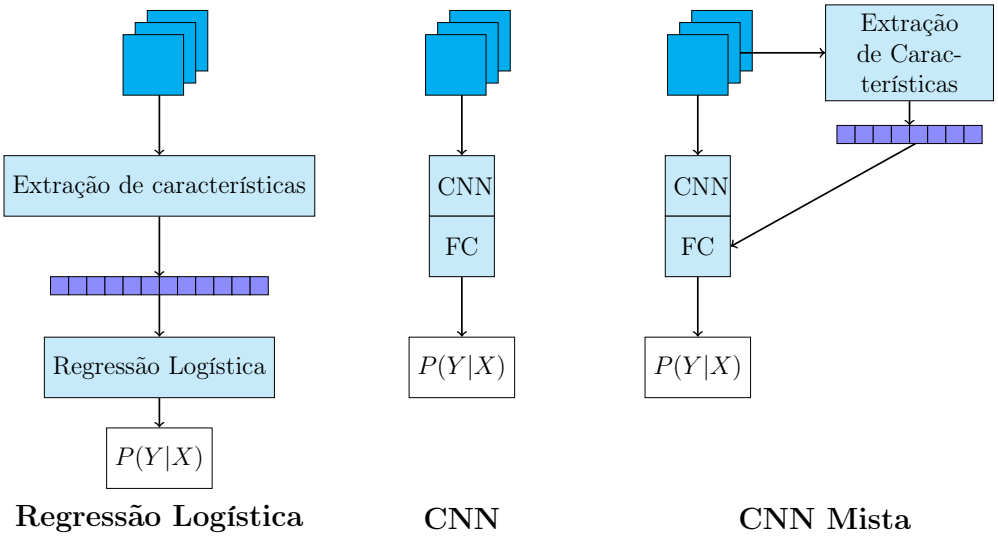

Figura 4.1: *Comparação dos três modelos estudados nesse trabalho*

4.2 Desenho Experimental

Neste trabalho implementamos os 3 modelos descritos previamente e os aplicamos em um problema real. Este problema é a classificação automática de plâncton em um dataset

produzido pelo LAPS semelhante ao usado nos últimos trabalhos da colaboração do IME e LAPS. Recebemos duas versões do dataset, uma original e uma com remoção de fundo (uma descrição mais detalhada do dataset e seu tratamento está na seção de experimentos).

O principal objetivo destes três experimentos é avaliar como os dois primeiros modelos funcionam separadamente e como se dá a interação entre eles agindo como partes do modelo misto. Queremos estudar se CNNs podem agir como extratores de características adicionais para melhorar os classificadores já existentes independentemente. Além disso, queremos avaliar a interação do conhecimento produzido por humanos (na forma das características extraídas e tratamento do dataset) com o extraído automaticamente por CNNs.

Para isso, fizemos três sets de experimentos, um em cada tipo de modelo estudado. Dividimos previamente um conjunto de treino e de teste em proporção 90%/10%, usando a acurácia de teste como indicador final de performance de cada modelo, e acompanhamos a função de custo pelo tempo em cada treinamento para acompanhar a velocidade de convergência e a presença de overfitting. Para os modelos em que precisamos otimizar hiperparâmteros também fizemos uma divisão adicional no dataset de treino em treino e validação.

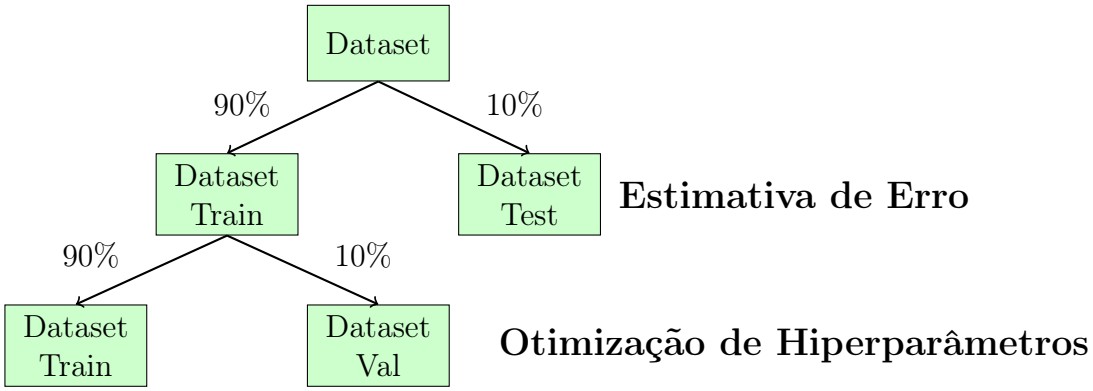

Figura 4.2: *Divisão de Dataset e propósito. Experimentos são feitos de baixo para cima*

No treino de CNNs, fizemos experimentos com o dataset com e sem extração de fundo. Usamos validação cruzada para determinar o tamanho de imagem ótimo antes de treinar e avaliar o desempenho no dataset de teste.

Para a regressão logística, usamos três segmentações (Otsu, Yen, Watershed), em cima das duas versões do dataset ao que tivemos acesso, gerando seis sets de features. Dado que uma regressão logística treinada com cada um dos sets é um modelo diferente, treinamos diretamente com o dataset de treino e testamos com o dataset de teste.

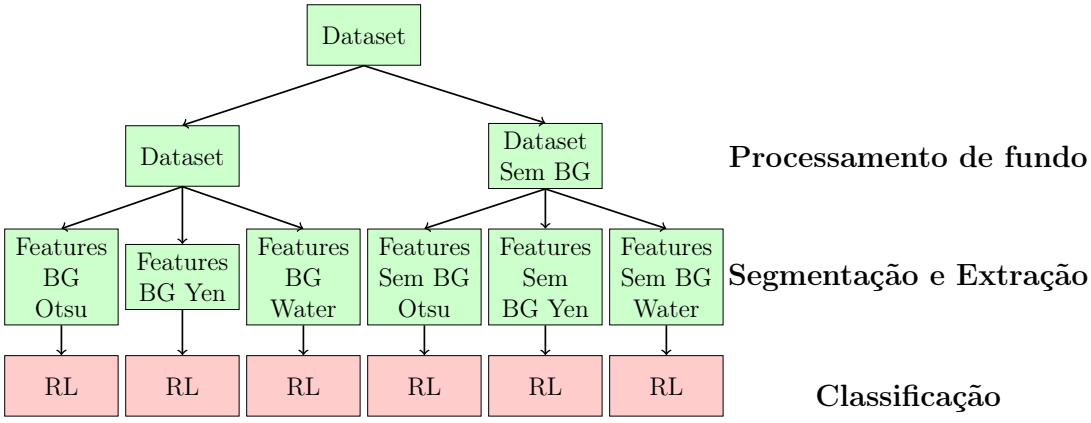

Figura 4.3: *Desenho Experimental da Regressão Logística*

O treino de CNNs mistas envolve desafios próprios que precisam ser enfrentados antes de comparar o desempenho com os outros modelos. Além da escolha de hiperparâmetros, o treino em si pode ser feito de diversas maneiras (ver seção de experimentos). Desse modo, primeiro escolhemos um conjunto de imagens e features quaisquer e o usamos para descobrir o melhor método para treinar as CNNs Mistas e otimizar os hiperparâmetros com validação cruzada. Em seguida extrapolamos os resultados encontrados nesta combinação dataset/features para os próximos experimentos.

Para evitar uma explosão combinatória de hiperparâmetros de CNN (número de filtros, tamanho de imagem), versões das imagens do dataset (com ou sem extração de fundo) e sets de features (seis), fizemos uma abordagem distinta para treinar CNNs Mistas: treinamos diretamente no dataset de treino seis modelos usando os seis sets de features e a versão do dataset utilizada para extrair essas features, e comparamos a acurácia de teste com a da regressão logística, de forma a concluir se houve ou não melhora.

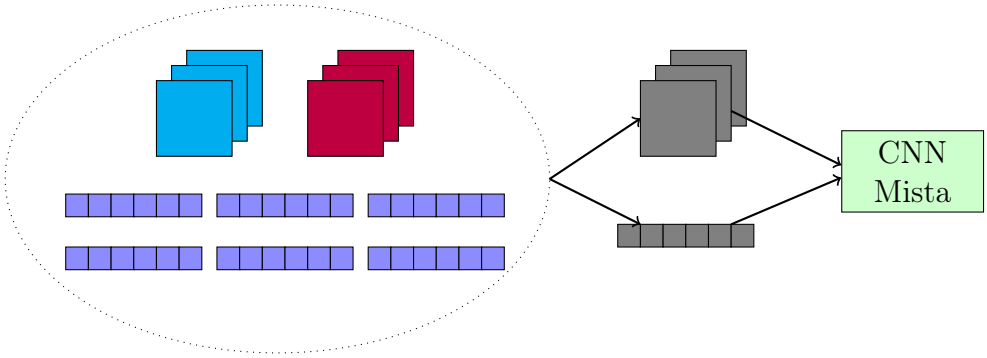

Figura 4.4: *Fase 1: combinação aleatória de imagens e features é escolhida para otimizar parâmetros*

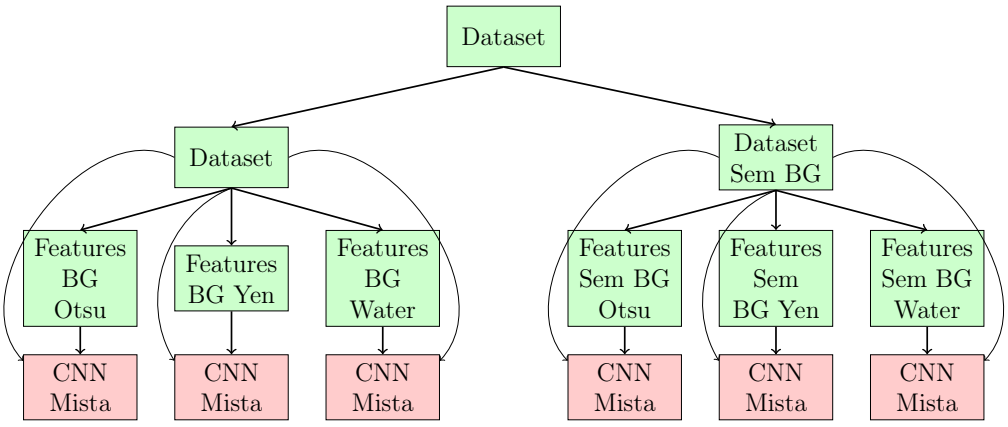

Figura 4.5: *Fase 2: experimentos feitos para espelhar os da Regressão Logística*

Capítulo 5

Experimentos

Para este trabalho, testamos três famílias de modelos: Redes Neurais Convolucionais, Regressão Logística sobre um set de características extraídas manualmente e Redes Neurais Convolucionais treinadas adicionalmente com o set de características (redes neurais mistas). Todos os modelos foram testados na mesma tarefa de classificação supervisionada, em um dataset proporcionado pelo LAPS. Acompanhamos para cada um a acurácia, velocidade de convergência do modelo, e presença de overfitting como medidas de performance e meio de comparação da informação aprendida.

5.1 Dataset

5.1.1 Aquisição e Seleção

As imagens foram adquiridas por um sistema de aquisição *in situ* desenvolvido pelo LAPS. O sistema coletou imagens submergido entre 1 a 30 metros em um laboratório em Ubatuba, na costa de São Paulo. Imagens foram capturadas a aproximadamente 15 frames por segundo, com dimensão de 2448×2050 e resolução $\sim 5\mu m$ e agrupadas em vídeos.

Dezesseis vídeos foram selecionados para a criação do dataset, gerando cerca de 230.000 regiões de interesse, das quais 6108 foram selecionadas para constituir o dataset. Um processo de classificação e anotação manual foi realizado por experts do laboratório. O dataset final possui 55 classes com número de exemplos entre 2 e 536. Abaixo exemplos de figuras do dataset.

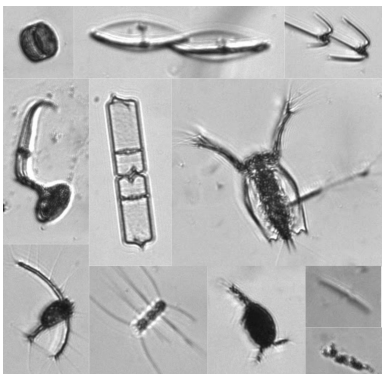

Figura 5.1: *Imagens pertencentes ao dataset. Cada imagem é de uma classe distinta*

Consideramos que o desbalanceamento entre classes, por mais que seja um problema tanto interessante como tratável, estaria mais adequado para ser tratado em futuros trabalhos.

Label	Class Name
0	cla_sp
1	cop_cal_sp
2	detritus_df_bk
3	detritus_uf_dot_bk
4	detritus_uf_dot_bw
5	detritus_uf_stick_bw
6	din_tri_sp
7	nauplii
8	phy_0_sp
9	phy_cha_sp

Tabela 5.1: *Classes e labels associados*

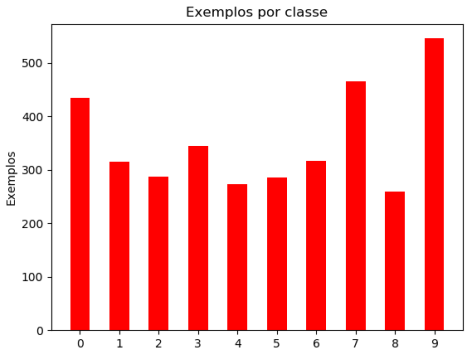

Figura 5.2: *Histograma de classes e número de exemplos*

Portanto selecionamos somente as 10 classes mais populosas para o dataset final. Acima temos uma tabela com as classes, seus labels associados e um histograma do número de exemplos.

5.1.2 Extração de Fundo

Imagens *in-situ* são suscetíveis a variabilidade natural em iluminação, turbulência e outros fatores que podem comprometer a qualidade e a classificação de exemplos, dado que regiões de interesse capturadas em vídeos distintos podem ser afetados por essa variabilidade (ver Figura 5.1).

Para tentar lidar com esse potencial problema, uma técnica de remoção de fundo adaptada para lidar com mudanças de iluminação (ver [18]) foi aplicada ao dataset. No entanto, como qualquer técnica deste tipo, ela pode ser destrutiva. Por isso, em todos os experimentos usamos ambas as versões e comparamos a performance.

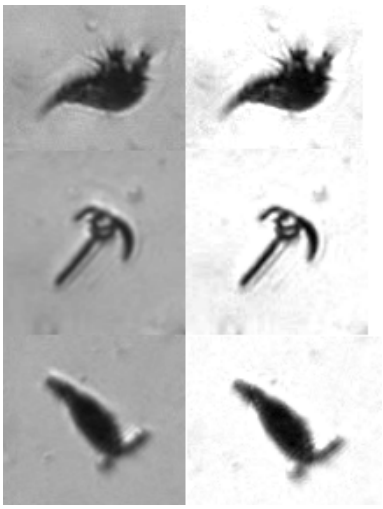

Figura 5.3: *Exemplos de aplicação de algoritmo de remoção de fundo*

5.1.3 Redimensionamento de Imagens

Como visto, CNNs tem um tamanho fixo de entrada. Logo há a necessidade de padronizar o tamanho de imagens antes do treino ou teste. O algoritmo utilizado para isso é descrito abaixo.

Algorithm 2 Redimensionamento de Imagens

```
1: procedure RESIZE(image, target)        ▷ Função que torna image.shape target×target
2:     maxdim ← max(image.shape)
3:     if maxdim > target then
4:         image ← DOWNSIZE(image, target/maxdim)
5:     image ← PAD(image, target)
```

DOWNSIZE(image, alpha) é uma função (*scipy.misc.imresize*) que diminui proporcionalmente uma imagem por um fator alpha.

PAD(image, target) é uma função (*np.pad*) que aumenta uma imagem ao adicionar zeros nas bordas de uma forma simétrica (zero-padding).

5.1.4 Aumentação de Dados

Para os modelos que recebem imagens como entrada, foi usado um esquema de aumentação de dados para aumentar a robustez e performance do classificador. Uma dentre seis transformações (identidade, flip horizontal, flip vertical, rotação em 90, 180 e 270 graus) é escolhida aleatoriamente e aplicada a cada batch.

5.2 Extração de Características

Seguindo o pipeline tradicional de classificação de imagens descrito na seção de fundamentos teóricos, as imagens foram segmentadas e dessa segmentação foram extraídas diversas características para compor um vetor de features. Tanto a escolha dos algoritmos de segmentação como das características extraídas foram bastante informadas por pesquisas anteriores surgidas da colaboração do IME com o IO-LAPS[14] [22], bem como pelo funcionamento dos sistemas presentes e em uso atualmente do LAPS.

As características escolhidas para extração foram as mesmas utilizadas pelo sistema atual do LAPS, e bem próximas das utilizadas pelas pesquisas anteriores. Traram-se de 54 características, sendo 14 momentos de imagem, 16 estatísticas de tons de cinza (histograma, média, entropia, curtose, entre outros) e 24 descritores de forma (circularidade, convexidade, entre outros).

5.3 Modelos de Classificação

Os modelos de classificação citados acima foram todos implementados em Lasagne [10], uma biblioteca primariamente de redes neurais em Python. No caso da regressão logística, usamos também a implementação do Scikit-Learn [25], baseada em métodos de otimização de segunda ordem, que tem melhor convergência.

5.4 Treino

Todos os modelos foram treinados com Adagrad[11], uma variante do método do gradiente que adapta a taxa de aprendizado a cada parâmetro, aumentando para parâmetros com pouca mudança e diminuindo para parâmetros com muita, dessa forma permitindo uma padronização maior dos experimentos e menos preocupação com hiperparâmetros.

Capítulo 6

Resultados

6.1 CNN

CNNs foram construídas de acordo com o algoritmo descritos na seção de fundamentos teóricos, com diferentes configurações de tamanho de imagem e número de filtros. Resultados foram compilados abaixo.

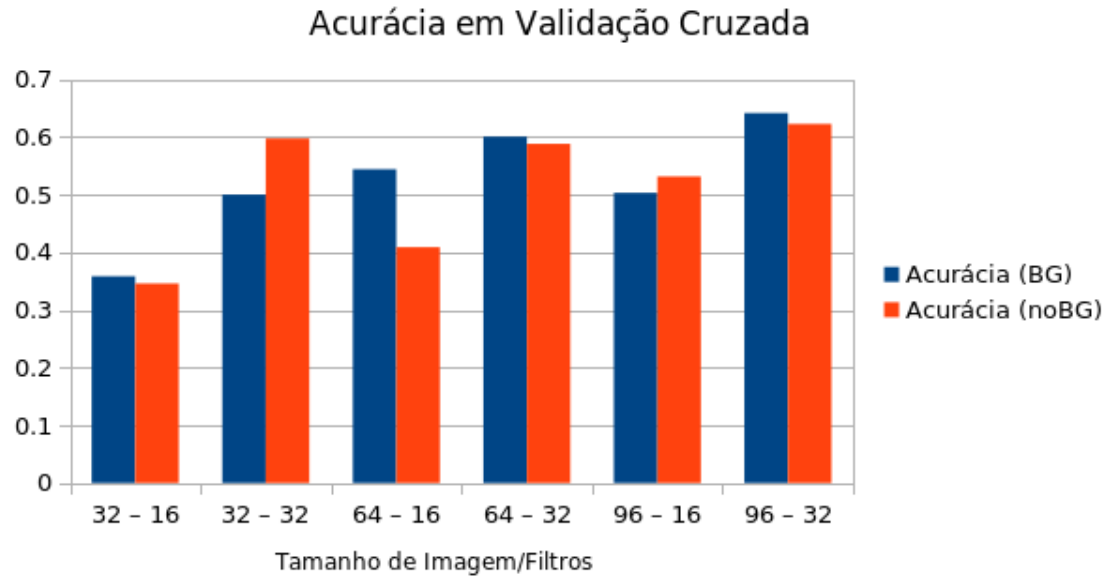

Figura 6.1: *Resultados da validação cruzada*

Percebe-se uma forte tendência entre tamanho de entrada e acurácia, mas principalmente entre o número de filtros e acurácia. Parece haver aí indicação do fato de que a generalização funciona diferentemente para Deep Learning. Apesar da acurácia de teste semelhante, o gráfico parece indicar uma interferência mais negativa do que positiva do algoritmo de extração de fundo.

A curva da função de erro, em geral, tem uma queda grande seguida de uma tendência bem mais suave, o que pode passar uma impressão equivocada de estacionariedade quando visto como um todo no gráfico. Além disso a curva é instável, possuindo alguns pulos, de modo que o uso de Early Stopping de maneira simples (parar se há aumento de uma época

para outra) não é informativo. Assim, há um problema de definir o ponto de parada/-quantidade de épocas para o treinamento, que é perceptível pela indicação de tendência a decrescimento nos gráficos. Testes com mais épocas, no entanto, revelaram que a acurácia conseguida nestes experimentos já são bem próximas do máximo possível.

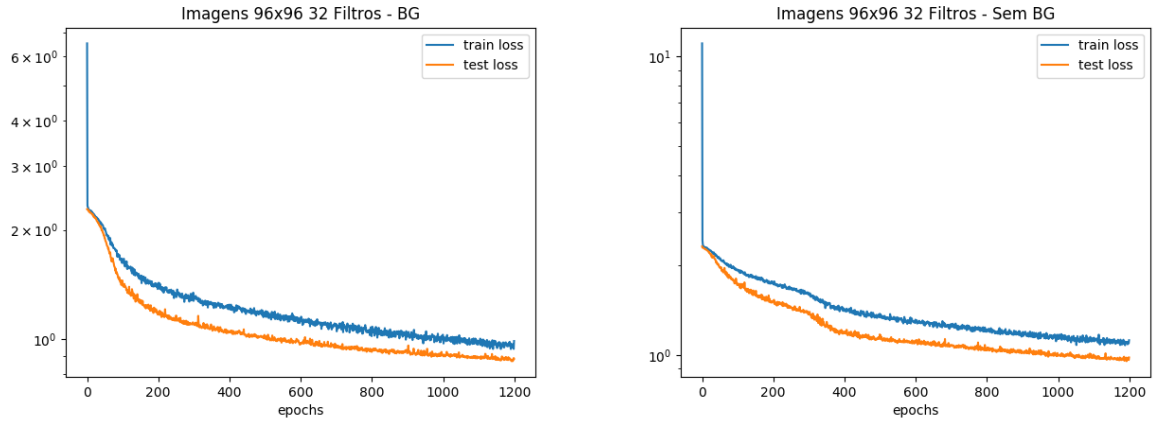

Figura 6.2: *Função de perda no tempo do teste*

Com Background	Sem Background
0.6940	0.6373

Tabela 6.1: *Acurácia de Teste - CNN*

6.2 Regressão Logística

A regressão logística teve uma convergência bem mais lenta que as CNNs, mesmo com uma taxa de aprendizado maior. A curva da função de perda em relação ao tempo também é mais suave:

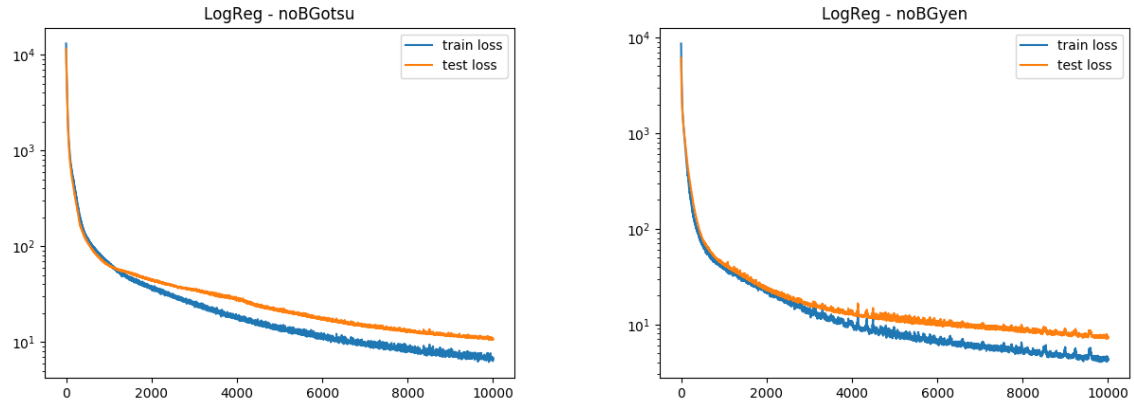

Figura 6.3: *Função de perda no tempo de dois experimentos*

Nota-se que, novamente, as curvas indicam que o treino ainda não convergiu para algum mínimo, havendo espaço para maior otimização. De fato, uma dificuldade constante durante os experimentos foi assegurar uma boa convergência para os modelos, sujeitos a grande variância entre testes por diferenças de inicialização e convergência. Não se observou o mesmo em experimentos com o pacote pronto Scikit-Learn, provavelmente devido à robustez dos métodos de otimização utilizados. Abaixo colocamos a acurácia final dos experimentos:

	Sem Background	Com Background
Otsu	0.6473	**0.7308**
Yen	0.6827	0.6572
Watershed	0.6671	0.6827

Tabela 6.2: *Acurácia- Regressão Logística implementada em Lasagne*

	Sem Background	Com Background
Otsu	**0.7807**	0.7643
Yen	0.7473	0.7490
Watershed	0.7552	0.7688

Tabela 6.3: *Acurácia - Regressão Logística em Scikit-Learn*

Vemos que há uma grande diferença entre os resultados das duas implementações, que é possível de assumir que se deva a questões de convergência. É ainda interessante notar que não é possível dizer claramente se a remoção de fundo foi danosa ou proveitosa para este algoritmo.

6.3 CNN Mista

6.3.1 Modo de Treinar

Primeiramente, é notável que a curva de treino quando ambas as partes são treinadas ao mesmo tempo é bem mais caótica, tornando difícil o diagnóstico do treino (ponto de parada ótimo, convergência ou não, entre outros). Por isso, se implementou além do treinamento normal funções que atualizavam a parte profunda ou "rasa"individualmente. Em uma primeira etapa, então, verificamos a diferença de realizar o treino todo com as duas partes ou separar cada parte, começando por cada parte.

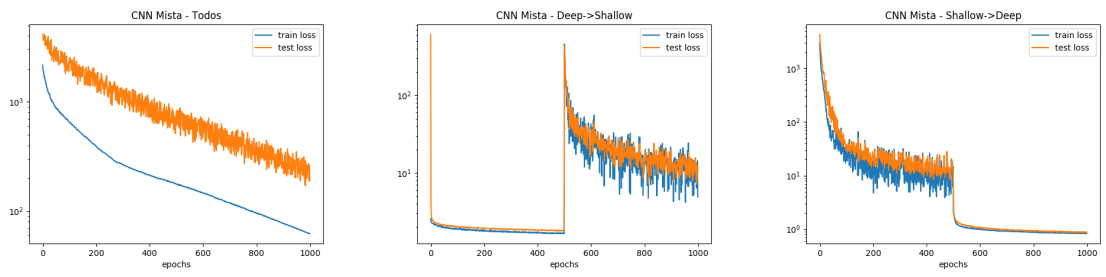

Figura 6.4: *Comparação de três métodos de treinamento*

Além da instabilidade maior já comentada de quando as partes são treinadas juntas, notamos que quando as partes são treinadas separadamente é bem clara a mudança entre uma parte e outra, cada parte tomando uma forma mais parecida com a do treino individual.

Percebemos também que treinar a parte profunda antes não traz resultados bons, o que faz sentido com a teoria: supomos que a CNN se adaptaria à informação presente das features, aprendendo a extrair informação distinta das imagens. Ao realizar o treino nesta ordem isso não pode acontecer, o que pode levar a uma redundância das duas partes. O fato de que a função de perda não chega a voltar aos níveis anteriores depois do salto na mudança de parte parece comprovar essa hipótese. O salto em si possivelmente se deve ao fato de que a parte profunda tem muito mais parâmetros, estando mais sensível a mudanças bruscas nas entradas, correspondendo às primeiras iterações do treino da parte rasa.

Dada a instabilidade do treino conjunto, a vantagem de treinar em duas etapas ajustando os hiperparâmetros para aproximar aos usados nas seções anteriores e o fato de que treinar conjuntamente ou primeiro a parte rasa não provocou mudança aparente na performance do modelo, optamos pelo terceiro método de treino para os próximos experimentos: treinar primeiro a parte rasa e depois a parte profunda.

6.3.2 Validação Cruzada

Uma vez determinada a maneira ótima de realizar o treino, fizemos validação cruzada sobre uma combinação aleatória de features e imagens para determinar a influência do tamanho da imagem e número de filtros sobre a performance.

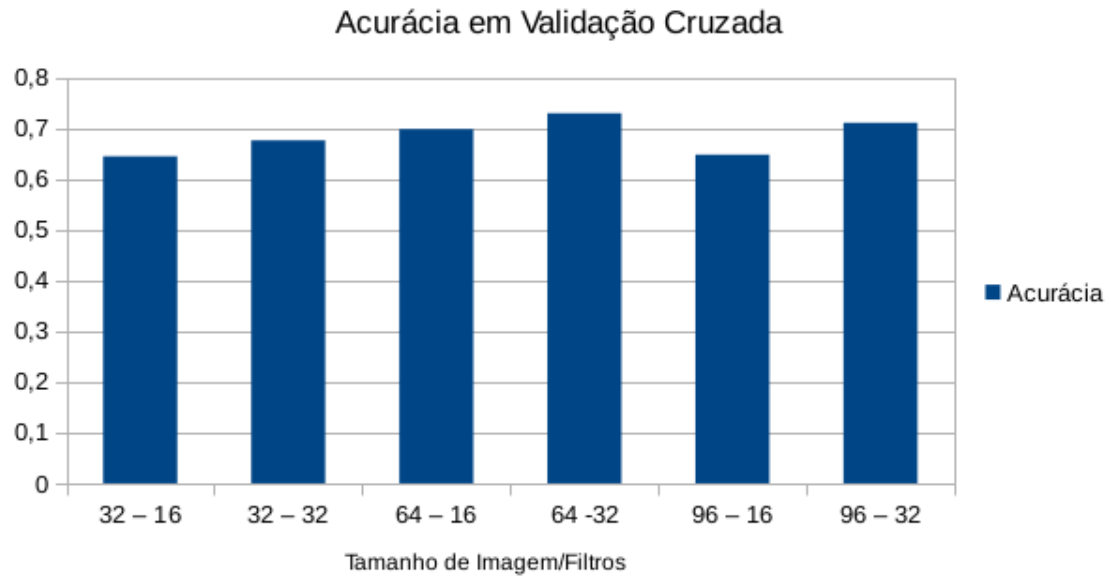

Figura 6.5: *Resultados da validação cruzada*

Vemos aqui novamente uma tendência de maior performance relacionada a maior número de parâmetros e tamanho de imagem, mas menos acentuada. Parece haver uma pequena queda após atingir o pico no modelo com imagens de 64 por 64 e 32 filtros. Apesar da queda não ser suficiente para atribuir a overfitting ou modelos complexos demais, já é o suficiente para determinar que este modelo será o utilizado nos próximos testes.

6.3.3 Teste e Comparação

Finalmente, com os dados dos dois últimos experimentos podemos fazer os experimentos comparativos:

Dataset	CNN	Regressão Logística	CNN Mista
Otsu/BG	0.6940	0.7308	**0.7393**
Otsu/sem BG	0.6373	0.6473	**0.7138**
Yen/BG	0.6940	0.6572	**0.7648**
Yen/sem BG	0.6373	0.6827	**0.7167**
Watershed/BG	0.6940	0.6827	**0.7592**
Watershed/sem BG	0.6373	0.6671	**0.6855**

Tabela 6.4: *Comparação de acurácia de teste dos três modelos*

Capítulo 7

Conclusões

É seguro concluir que os objetivos deste trabalho foram alcançados: conseguimos implementar um modelo misto e desenvolver técnicas para treiná-lo eficientemente. Os modelos propostos consistentemente superaram os modelos individuais de CNN e Regressão Logística, por vezes com uma margem de até 11 pontos percentuais na acurácia (Watershed/BG).

Além disso, confirmamos vários postulados teóricos conhecidos: CNNs precisam de datasets maiores para obter um bom resultado e o número de parâmetros não parece estar ligado a esta necessidade, de modo que diminuir o número de parâmetros (filtros) não ajuda a mitigar este efeito. Interessantemente, no entanto, no caso de redes neurais mistas a escolha ótima não foi o modelo com mais parâmetros.

Em aspectos mais práticos, podemos concluir que, no geral, a técnica de extração de fundo utilizada foi mais danosa do que proveitosa, embora isso seja mais pronunciado para CNNs e menos para extração de características. É interessante notar como a diferença é mais pontuada nas CNNs mistas, o que parece indicar que parte das características alternativas extraídas pelas CNNs estão nos artefatos removidos pela extração de fundo. Além disso, é bem seguro concluir que a segmentação de Otsu foi a que produziu melhores resultados entre as utilizadas.

7.1 Futuros Desenvolvimentos

Dado o sucesso neste domíno, a aplicação da técnica desenvolvida neste trabalho em outros datasets e outros domínios pode consolidar a ideia de CNNs mistas como uma maneira genérica de melhorar classificadores.

Experimentos comparando os resultados obtidos neste trabalho com outras formas mais simples de combinação de modelos (Ensembling, Bagging) podem trazer mais força à hipótese de que as CNNs Mistas aprendem a extrair informações distintas do que CNNs normais. Além disso, comparações das CNNs normais e mistas por meio de, por exemplo, análise qualitativa dos filtros aprendidos em cada caso, podem revelar mais sobre o tipo de informação aprendida por CNNs e a relação com a informação contida nas features.

A principal dificuldade encontrada no trabalho foi definir o número de épocas de treinamento, bem como saber o momento de interromper o treino prematuramente. O estudo e desenvolvimento de métodos de parada mais complexos e mais robustos a alta variabilidade se faz necessário, então.

Bibliografia

[1] Abu-Mostafa, Y. S., Magdon-Ismail, M., and Lin, H.-T. *Learning From Data*. AMLBook, 2012. 5

[2] Al-Shabi, M., Cheah, W. P., and Connie, T. Facial expression recognition using a hybrid CNN-SIFT aggregator. *CoRR abs/1608.02833* (2016). 4

[3] Antipov, G., Berrani, S.-A., Ruchaud, N., and Dugelay, J.-L. Learned vs. hand-crafted features for pedestrian gender recognition. In *Proceedings of the 23rd ACM International Conference on Multimedia* (New York, NY, USA, 2015), MM '15, ACM, pp. 1263–1266. 4

[4] Bengio, Y. Deep Learning of Repr. for Unsupervised and Transfer Learning. In *Proc. of ICML Work. on Unsuperv. and Transf. Learning* (2012), pp. 17–36. 4

[5] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 7

[6] Blaschko, M. B., Holness, G., Mattar, M. A., et al. Automatic in situ identification of plankton. In *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on* (Jan 2005), vol. 1, pp. 79–86. 2, 3

[7] Cowen, K., R., Sponaugle, S., Robinson, K., and Luo, J. PlanktonSet 1.0: Plankton imagery data collected from F.G. Walton Smith in Straits of Florida from 2014-06-03 to 2014-06-06 and used in the 2015 National Data Science Bowl. 4

[8] Dai, J., Wang, R., Zheng, H., Ji, G., and Qiao, X. ZooplanktoNet: Deep Conv. Network for Zooplankton Classification. In *OCEANS 2016 - Shanghai* (April 2016), pp. 1–6. 4

[9] Dieleman, S., Fauw, J. D., and Kavukcuoglu, K. Exploiting cyclic symmetry in convolutional neural networks. *CoRR abs/1602.02660* (2016). 3, 4

[10] Dieleman, S., Schlüter, J., Raffel, C., et al. Lasagne: First release., Aug. 2015. 19

[11] Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization, 2010. 8, 20

[12] Ellen, J., Li, H., and Ohman, M. D. Quantifying california current plankton samples with efficient machine learning techniques. In *OCEANS 2015 - MTS/IEEE Washington* (Oct 2015), pp. 1–9. 3

[13] Fernandes, J. A., Irigoien, X., Boyra, G., Lozano, J. A., and Inza, I. Optimizing the number of classes in automated zooplankton classification. *Journal of Plankton Research 31*, 1 (2009), 19–29. 3

[14] Fernandez, M. A. Classificação de imagens de plâncton usando múltiplas segmentações. 2, 3, 9, 19

[15] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 3, 10

[16] Gorsky, Gaby, Ohman, et al. Digital zooplankton image analysis using the zooscan integrated system. *Journal of Plankton Research 32*, 3 (2010), 285–303. 2, 3

[17] Henson, S., Sanders, R., and Madsen, E. Global patterns in efficiency of particulate organic carbon export and transfer to the deep ocean. 1028–. 2

[18] Jacques, J. C. S., Jung, C. R., and Musse, S. R. A background subtraction model adapted to illumination changes. *Proceedings - International Conference on Image Processing, ICIP* (2006), 1817–1820. 18

[19] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep conv. neural networks. In *Adv. in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105. 1, 3

[20] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (November 1998), 2278–2324. 1, 3

[21] LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural Networks: Tricks of the Trade* (London, UK, UK, 1998), Springer-Verlag, pp. 9–50. 3, 8

[22] Matuszewski, D. J. Computer vision for continuous plankton monitoring. 2, 3, 19

[23] Orenstein, E. C., and Beijbom, O. Transfer learning and deep feature extraction for planktonic image data sets. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (March 2017), pp. 1082–1088. 4

[24] Pan, S. J., and Yang, Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering 22*, 10 (Oct 2010), 1345–1359. 4

[25] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830. 19

[26] Py, O., Hong, H., and Zhongzhi, S. Plankton classification with deep convolutional neural networks. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference* (May 2016), pp. 132–136. 4

[27] Rodrigues, F. C. M., Hirata, N. S. T., Abello, A. A., Cruz, L. T. D. L., Lopes, R. M., and Jr, R. H. Evaluation of transfer learning scenarios in plankton image classification. In *To appear in proceedings of VISAPP 2018* (2018). 4

[28] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014). 3

[29] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Sutskever, Ilya, Salakhutdinov, and Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res. 15*, 1 (Jan. 2014), 1929–1958. 12

[30] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *CoRR abs/1409.4842* (2014). 3

[31] Wang, H., Cruz-Roa, A., Basavanhally, A., Gilmore, H., et al. *Cascaded ensemble of convolutional neural networks and handcrafted features for mitosis detection*, vol. 9041. SPIE, 2014. 4

[32] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Adv. in neural information processing systems* (2014), pp. 3320–3328. 4

[33] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *CoRR abs/1611.03530* (2016). 11

[34] Álvarez, E., Lopez-Urrutia, A., and Nogueira, E. Improvement of plankton biovolume estimates derived from image-based automatic sampling devices: Application to flowcam. 454–469. 2, 3