TRABALHO DE GRADUAÇÃO

SEGMENTAÇÃO SEMÂNTICA DE ALIMENTOS BRASILEIROS: UMA ABORDAGEM UTILIZANDO MODELOS DE APRENDIZAGEM PROFUNDA

Bruno Fernandes Carvalho

Tomás Dias de Queiroz

Brasília, outubro de 2021

ENGENHARIA MECATRÔNICA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

SEGMENTAÇÃO SEMÂNTICA DE ALIMENTOS BRASILEIROS: UMA ABORDAGEM UTILIZANDO MODELOS DE APRENDIZAGEM PROFUNDA

Bruno Fernandes Carvalho

Tomás Dias de Queiroz

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Flávio de Barros Vidal, CIC/UnB _____
Orientador

Prof. Adolfo Bauchspiess, ENE/UnB _____
Examinador interno

Prof. Marcus Vinicius Lamar, CIC/UnB _____
Examinador interno

Brasília, outubro de 2021

FICHA CATALOGRÁFICA

TOMÁS, DIAS DE QUEIROZ BRUNO, FERNANDES CARVALHO
Segmentação semântica de alimentos brasileiros - Uma Abordagem Utilizando Modelos de Aprendizagem Profunda

[Distrito Federal] 2021.

xvii, 67p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2021). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1. Redes Neurais Profundas 2. Segmentação Semântica de Alimentos
3. Processamento de Imagens 4. Food Computing

I. Mecatrônica/FT/UnB II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

QUEIROZ, TOMÁS. D. CARVALHO, BRUNO FERNANDES, (2021). Segmentação semântica de alimentos brasileiros - Uma Abordagem Utilizando Modelos de Aprendizagem Profunda. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-n°004, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, .

CESSÃO DE DIREITOS

AUTORES: Tomás Dias de Queiroz e Bruno Fernandes Carvalho

TÍTULO DO TRABALHO DE GRADUAÇÃO: Segmentação semântica de alimentos brasileiros - Uma Abordagem Utilizando Modelos de Aprendizagem Profunda.

GRAU: Engenheiro ANO: 2021

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Os autores reservam outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito dos autores.

Tomás Dias de Queiroz
SQS 306 Bloco I, Apartamento 506, Bairro Asa Sul - 70353-090 Brasília – DF – Brasil.

Bruno Fernandes Carvalho
SMAS Trecho 1 C Bl. A Apt. 508 Living Parksul - 71218-010 Brasília – DF – Brasil.

Dedicatórias

Dedico este trabalho primeiramente aos meus pais que me deram uma criação que me tornou capaz de aprender conhecimentos de base tecnológica sem perder a empatia e o lado humano. Em segundo lugar dedico ao meu querido amigo Bruno com o qual tenho amizade de mais de uma década e com quem desejo colher muitos frutos tanto profissionais quanto existenciais.

Tomás Dias de Queiroz

Dedico este trabalho a todos os diabéticos que não tiveram o mesmo privilégio de tratamento e acesso à informação como eu tive. Esse projeto é um pontapé inicial para incluir e democratizar tecnologias que incentivem o autocuidado e auxiliem o gerenciamento da diabetes. .

Bruno Fernandes Carvalho

Agradecimentos

Agradeço a Sandra, Clara, Eliseu, Halexya e amigos pela amizade e por terem me ajudado a manter minha saúde mental durante longos anos de estudo. Sem a educação que tive e o suporte emocional, não seria possível estar concluindo esse trabalho. Especial agradecimento ao amigo Tomás por ter me influenciado a fazer Engenharia Mecatrônica, por ter sido um companheiro de projetos durante toda a Graduação e por ter sempre apoiado e incentivado minhas iniciativas. Também fico grato pela Universidade de Brasília ter investido na minha formação profissional e espero retribuir esse privilégio melhorando nossa sociedade.

Bruno Fernandes Carvalho

Agradeço a mim mesmo pelas horas de luta e trabalho árduo para finalizar essa Graduação.

Tomás Dias de Queiroz

RESUMO

A necessidade de monitorar e direcionar uma alimentação saudável às pessoas é urgente, vista a enorme quantidade de pessoas com doenças e condições que podem ser causadas, controladas e solucionadas a partir da alimentação. Com esta motivação, e com a possibilidade da utilização massiva de redes neurais profundas para segmentar e classificar alimentos em uma imagem, abrem portas para inúmeras aplicações na área de Computação voltada à alimentos (*Food Computing*). Tarefas como monitoramento de dieta e identificação de hábitos alimentares são passíveis de serem realizadas com auxílio do uso destas redes. Desta feita, neste trabalho é apresentado um banco de dados próprio, focado exclusivamente em alimentos brasileiros, visto que a diferença cultural alimentar do Brasil em relação a outros países inviabiliza a utilização de bancos de dados de alimentos de outras culturas. Foram utilizadas técnicas de processamento de imagens e redes neurais profundas para realizar a segmentação e classificação destes alimentos. A implementação proposta também superou trabalhos de estado-da-arte em pelo menos 4% a mais pela métrica da média da Interseção sobre a União (*Mean Intersection-Over-Union - mIoU*), se comparado com outros modelos e base de dados disponíveis na literatura.

Palavras Chave: Redes Neurais Profundas, Segmentação Semântica de Alimentos, Processamento de Imagens, *Food Computing*, Visão Computacional, Aprendizado de Máquina

ABSTRACT

The need to monitor and direct people to healthy eating is urgent, given the huge number of people with diseases and conditions that can be caused, controlled, and resolved through eating. With this motivation, and with the possibility of massive use of deep neural networks to segment and classify foods in an image, they open doors for numerous applications in the field of Computing focused on foods (*Food Computing*). Tasks such as monitoring diet and identifying eating habits are likely to be carried out with the help of the use of these networks. This time, this work presents its own database, focused exclusively on Brazilian foods, as the cultural differences between Brazil and other countries make it impossible to use databases of foods from other cultures. Image processing techniques and deep neural networks were used to classify foods using images of these foods as a basic principle. The proposed implementation also outperformed state-of-the-art works by at least 4% more by the metric of the mean Intersection over the Union (*Mean Intersection-Over-Union - mIOU*), compared to other models and datasets available in the literature.

Keywords: Deep Neural Networks, Food Semantic Segmantation, Image Processing, Food Computing, Computer Vision, Machine Learning

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

API	Interface de programação de aplicações – *Application Programming Interface*.
CNN	Rede Neural Convolucional – *Convolutional Neural Network*.
CPU	Unidade de Processamento Central – *Central Processing Unit*.
GPU	Unidade de Processamento Gráfico – *Graphics Processing Unit*.
UnB	Universidade de Brasília.
CIC	Ciência da Computação.
IOU	Interseção sobre União – *Intersection over Union*.
mIOU	Média de Interseção sobre União – *Mean Intersection over Union*.
SGD	Gradiente Descendente Estocástico – *Stochastic Gradient Descent*.
SVM	Máquina de Vetores de Suporte –*Support Vector Machine*.
LR	*Learning Rate*.
GT	Verdade de referência – *Ground Truth*.
DHASH	*Difference Hashing*.
KFOLD	*Cross Validation*.
UECFoodPix	*Dataset* de segmentação semântica do artigo [20].
FoodSeg103	*Dataset* de segmentação semântica do artigo [3].
UNIMIB16	*Dataset* de segmentação semântica do artigo [16].
MyFood	*Dataset* de segmentação semântica do artigo [21].
BRUMAS21	*Dataset* criado com alimentos brasileiros [24].
SPRINT0	Cenário de testes com apenas 77 classes.
SPRINT1	Cenário de testes com apenas 17 classes.
SPRINT2	Cenário de testes com apenas 5 classes.

Capítulo 1

Introdução

1.1 Contextualização

Food Computing [1] é uma subárea de conhecimento que utiliza métodos da ciência da computação para realizar estudos relacionados ao alimento. Os estudos nessa área envolvem a aquisição e análise de dados de alimentos, utilizando-se de recursos como visão computacional, aprendizado de máquinas, mineração de dados e outros adventos tecnológicos com o intuito de prover serviços para melhorar a saúde humana, guiando o comportamento humano e entendendo a cultura relacionada.

Mais detalhadamente, *Food Computing* [1] pode ser dividida em 4 grandes etapas que podem ser visualizadas na Figura 1.1:

- Aquisição de dados: informações alimentícias são capturadas de diferentes formas (por exemplo, imagens de alimentos, registros de alimentos, receita, sabor e cheiro) a partir de diferentes fontes (por exemplo, redes sociais, IoT, mecanismos de busca e sites de compartilhamento de receitas)
- Análise de dados: é necessário extrair e processar algo útil dos dados coletados. Assim, técnicas de Ciência de Dados, Visão Computacional, Processamento de Linguagem Natural, Aprendizagem de Máquina e Mineração de Dados são utilizadas para transformar um dado cru em algo relevante e que agregue valor ao ser humano. Além da ciência da computação, esta área de conhecimento também toma emprestado teorias e métodos de outras disciplinas, tais como neurociência, ciência cognitiva, psicologia e química.
- Tarefas: após o dado ter sido devidamente processado e analisado, é possível realizar cinco tarefas básicas, como percepção, reconhecimento, extração, recomendação e predição e monitoramento.
- Aplicações: dado o resultado das tarefas, é permitido empregá-lo em diferentes contextos, como saúde, cultura, agricultura, medicina e ciência dos alimentos.

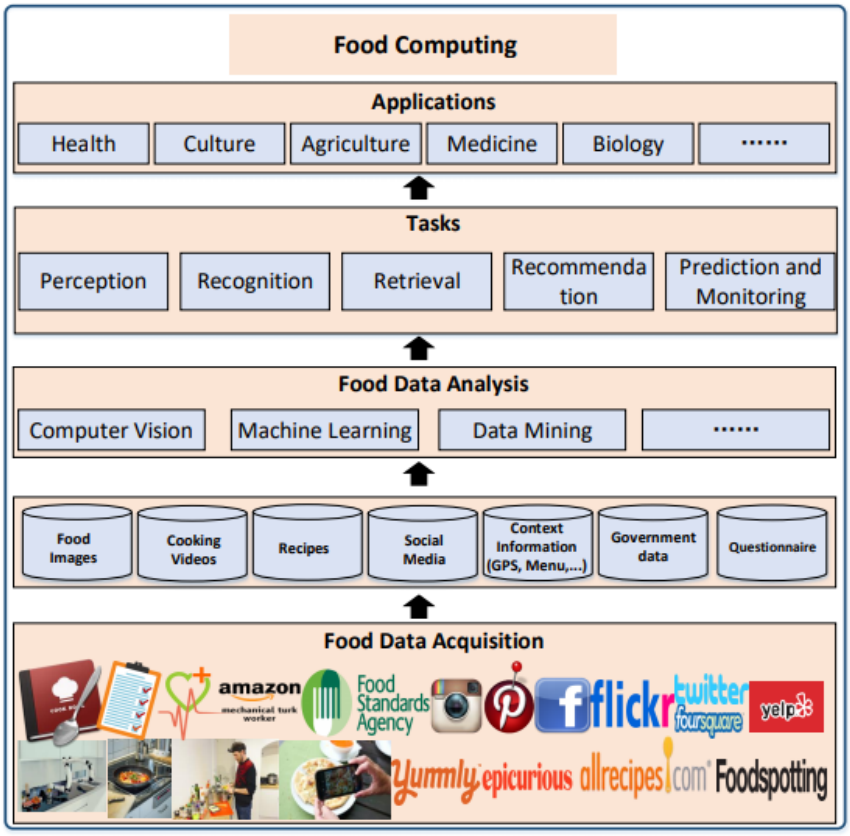


Figura 1.1: Visão geral de Food Computing [1].

Diversas aplicações recentes, como a recomendação de alimentos saudáveis para melhorar o hábito alimentar de pessoas [25], a classificação de imagens de alimentos utilizando aprendizado profundo [26], estimação de carboidratos para auxiliar na contagem de carboidrato [27], estimação de calorias a partir de imagens de alimentos [28] e a recomendação de receitas para indicar alimentos a partir dos ingredientes na foto [4] mostram a relevância e as diferentes possíveis abordagens nessa área.

Conectando os estudos mencionados e este trabalho, todos tem um aspecto em comum: são sistemas capazes de extrair informações de uma imagem usando técnicas de Inteligência Artificial e Visão Computacional. A partir do reconhecimento automático dos alimentos e da posição deles na imagem, é possível realizar várias tarefas, como a cálculo dos macronutrientes para avaliação da dieta ou até estimação da dose de insulina para diabéticos, baseando-se no cálculo da quantidade de carboidratos. São inúmeras aplicações que podem agilizar e facilitar a vida do ser humano: o *Survey* [1] resume bem esses aspectos do mundo de *Food Computing* e mostra as diferentes aplicações e tecnologias que estão sendo desenvolvidas.

1.1.1 Aplicações

1.1.1.1 Área da saúde

A alimentação é parte integrante de nossa vida. O que e o quanto comemos afeta de forma crucial nossa saúde. Por exemplo, se comemos demais, podemos correr o risco de desenvolver muitas doenças, tais como diabetes e doenças cardíacas. Por isso, estudos que compreendem a alimentação de uma pessoa beneficiará várias aplicações orientadas para a saúde. Existem quatro grupos representativos para aplicações em saúde, incluindo (1) percepção alimentar para a saúde, (2) reconhecimento alimentar para a gestão de dietas, (3) recomendação de alimentos e (4) análise de saúde alimentar das mídias sociais.

Nessa área, muitos trabalhos se concentram na estimativa de calorias a partir de uma imagem. Dentre os estudos mais relevantes e recentes desse tópicos, [29] forneceu o reconhecimento preciso de alimentos e medição de calorias através do treinamento periódico do sistema com mais imagens de alimentos. [30] propuseram o sistema Im2Calories, que utilizou uma abordagem baseada na segmentação para localizar a região de refeições da foto dos alimentos, e depois aplicou os classificadores multilabel para rotular essas regiões segmentadas. Uma vez o sistema segmentou os alimentos, ele pode estimar seu volume.

1.1.1.2 Cultura

A alimentação é fundamental para a cultura, com práticas alimentares que refletem nossas nacionalidades e outros aspectos relevantes. A compreensão disso é também indispensável na comunicação humana e não é verdade somente para profissionais de muitas áreas, como saúde pública e serviços de alimentação comercial, mas é claramente reconhecido no mercado global. Os alimentos também passaram a ser reconhecidos como parte da cultura local que os turistas consomem, como um elemento de promoção do turismo regional e um componente potencial da agricultura local e desenvolvimento econômico. Além disso, a exploração da cultura alimentar pode ajudar a desenvolver formas de recomendação de alimentos, considerando o aspecto cultural de diferentes áreas urbanas.

Por estas razões, cada vez mais trabalhos se concentram no estudo das culturas culinárias. [31] introduziu uma rede de sabores a partir de receitas para identificar uma série de padrões estatisticamente significativos que caracterizam a forma como os humanos escolhem os ingredientes das suas refeições. Os padrões se manifestam em graus variados em diferentes regiões geográficas. Por exemplo, os padrões norte-americanos e Os pratos da Europa Ocidental tendem a combinar ingredientes que compartilham o mesmo sabor.

1.1.1.3 Agricultura

A computação alimentar também pode ser usada na agricultura ou em produtos alimentícios. A análise da imagem dos alimentos tem muitas aplicações potenciais para tarefas automatizadas de agricultura e segurança alimentar. É possível, por exemplo, usar métodos de aprendizado profundo para extrair características visuais para a contagem de frutas ou legumes. Já [32] forne-

ceu uma visão geral das configurações de imagem hiper espectral e modos de detecção comuns para avaliação da qualidade e segurança alimentar.

Esse controle de qualidade envolve causas, prevenção e comunicação que tratam de doenças de origem alimentar. Alguns trabalhos [33] utilizaram a rede neural para automatizar a classificação da maturação do tomate e [34] propôs uma rede neural para distinguir com precisão entre ovos de grau A e ovos com manchas de sangue.

1.1.1.4 Ciência dos alimentos

É definida como a aplicação das ciências básicas e da engenharia para o estudo das ciências físicas, químicas e natureza bioquímica dos alimentos e os princípios do processamento de alimentos. *Food Computing* fornece novos métodos e tecnologias para estas sub-áreas: por exemplo, a análise sensorial consiste em estudar como os sentidos dos consumidores percebem os alimentos. Há estudo [35] que considerou este problema como o reconhecimento de alimentos em 1,9 milhões de imagens do *Instagram* e mostrou a lacuna de percepção entre como uma máquina rotularia objetivamente uma imagem e como um humano subjetivamente o faz.

Além disso, a percepção alimentar deve ser multimodal e inclui paladares, sabores, cheiros e sensações táteis. Portanto, a integração multimodal é necessária e estudos já existentes [36] focalizaram este tópico da neurociência e recorreram a um aprendizado profundo multimodal para melhor enfrentar este problema.

1.1.2 Tarefas em Food Computing

1.1.2.1 Percepção

Um aspecto importante que rege nossas escolhas alimentares e o quanto consumimos é como percebemos certas características de alimentos, como por exemplo, se são doces ou deliciosos. Portanto, o estudo sobre a percepção dos alimentos desempenha um papel importante para a nossa saúde. Além disso, tal estudo tem uma série de oportunidades para as indústrias de alimentos e bebidas, que podem ter uma melhor compreensão do processo utilizado pelas pessoas para avaliar a aceitabilidade e o sabor de novos produtos alimentícios.

Muitos estudos se concentram na percepção dos alimentos ao nível de atividade cerebral, tipicamente em laboratórios. Em resumo, pesquisas em percepção alimentar tem crescido bastante, especialmente na neurociência em campos relacionados à cognição e à saúde.

1.1.2.2 Reconhecimento

O uso generalizado de *smartphones* e os avanços de Visão Computacional permitiram o desenvolvimento de novos sistemas de reconhecimento de alimentos para o gerenciamento alimentar. Uma vez que reconhecemos a categoria ou ingredientes da refeição, nós podemos realizar várias análises relacionadas à saúde, como por exemplo, estimativa de ingestão calórica, análise nutricional e até mesmo o análise dos hábitos alimentares das pessoas. Além disso, o reconhecimento dos alimentos diretamente das imagens também é altamente desejável para outros contextos,

como em restaurantes *self-service*, onde o reconhecimento de alimentos poderia permitir tanto o monitoramento do consumo e o faturamento automático da refeição. Finalmente, para as pessoas que desejam obter uma melhor compreensão dos alimentos que elas não conhecem ou que nunca viram antes, elas podem simplesmente tirar uma foto e obter mais informações.

Devido ao grande número de aplicações e por ser uma tarefa básica em quase todo sistema de *Food Computing*, houve uma explosão de estudos nos últimos anos que envolvem o reconhecimento de alimentos, sendo estes divididos em três grandes grupos: Classificação, Detecção e Segmentação Semântica. Este trabalho se encaixa justamente nessa tarefa de *Food Computing*, pois acredita-se que ela é uma base para qualquer outra aplicação. A partir do entendimento dos alimentos presentes na imagem, é possível transformar esse dado em outras informações que também agregam valor, como dado nutricional ou detecção de hábito alimentar.

Classificação de Imagem

Existem duas possibilidades: classificação de imagens *Single-label* e classificação *Multi-label*. A primeira é responsável por atribuir apenas uma classe (alimento) para toda a imagem. Já a segunda pode atribuir inúmeras classes por imagem, porém sem indicar a posição dos alimentos. Isso pode ser bastante útil em aplicações que querem apenas entender o que a pessoa está comendo, sem saber a quantidade.

Um dos estudos mais relevantes feitos nessa área é o Food-101 [37], que contém 101 tipos de alimentos diferentes com 1000 imagens cada, sendo estes predominantemente ocidentais. Apesar do grande número de imagens e da grande variedade de ambientes, ruído e luminosidade, esta tarefa se encaixa na categoria de Classificação *Single Label*, onde há apenas um alimento por imagem (a tarefa se torna um pouco menos complexa). A figura 1.2 mostra algumas fotos desse estudo.

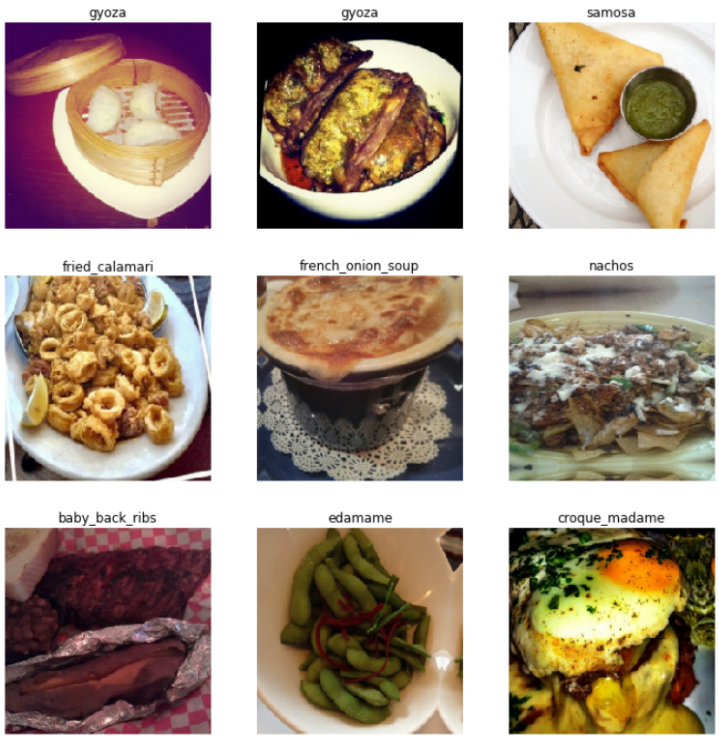


Figura 1.2: Classificação de imagens que contém apenas uma classe de alimentos.

Qualquer algoritmo de classificação de imagens se encaixa bem para resolver essa tarefa. Recentemente, a utilização de modelos de aprendizagem profunda são preferidos pela alta performance e pelo uso de *Transfer Learning*. Normalmente, treina-se um classificador com uma base de dados gigantesca com inúmeras classes como o *ImageNet* [38], com o objetivo de aprender como extrair características relevantes de uma imagem e que podem ser usadas posteriormente para discriminar entre diferentes classes. Após esse treinamento, é possível utilizar o conhecimento aprendido na base de dados grande e transferí-lo para outro contexto e outro *Dataset*, como o caso do Food-101.

Um modelo classificador é caracterizado por duas etapas: extração de características usando Redes Neurais Convolucionais Profundas (CNNs) e uma rede neural de conexão completa (*Fully Connected Layer*) para classificar a imagem. A primeira parte também é normalmente chamada de *encoder*, pois codifica as informações da imagem e dá de entrada para a cabeça classificadora de conexão completa. Exemplos de algoritmos que realizam essa tarefa nos últimos anos de forma eficiente são RESNET [39] e VGG [40], que possuem arquiteturas voltadas para esse tipo de tarefa.

Detecção de Objeto

Detectar objetos em uma imagem é extremamente relevante para diversas aplicações, como detecção de pessoas, rostos, carros etc. Uma dessas aplicações que explodiu nos últimos anos é o carro autônomo, que necessariamente opera usando Inteligência Artificial para entender a cena.

No contexto de *Food Computing*, cada objeto é uma classe de alimento diferente e é atribuído um retângulo (*bounding box*) a cada um eles, para identificar sua posição na imagem.

Um dos primeiros estudos com *Dataset* relevante foi o [41] UECFOOD100, que foi anotado especificamente para essa tarefa de detecção de objetos. Um exemplo pode ser visto na figura 1.3.

Figura 1.3: Detecção de alimentos na imagem [2].

Para esse estudo, normalmente são usadas CNNs para extrair as características dos objetos na imagem e depois é usado uma cabeça classificadora (rede neural com conexão completa) para indicar a classe do alimento e outra cabeça para estimar a posição do objeto, a partir da regressão dos dois pontos que constituem um retângulo. Existem diversos algoritmos de *Deep Learning* baseados em redes CNN prontos para realização dessa tarefa, como o YOLO [42] e o Faster-RCNN [43].

Segmentação Semântica

Muito similar à detecção de objetos, segmentação semântica também classifica objetos na imagem e indica a posição dos mesmos. A principal diferença é que a posição do objeto é dada de forma precisa, pixel a pixel, ao invés de apenas definir um retângulo com as coordenadas. Com um exemplo visual fica fácil de entender. Na figura 1.4, observa-se que as fronteiras dos alimentos são exatas: cada cor representa uma classe diferente de alimento e o fundo da imagem é uma classe que usa sempre a mesma cor. Outra forma de pensar nessa questão é que cada alimento e o fundo são representados por *ids* diferentes.

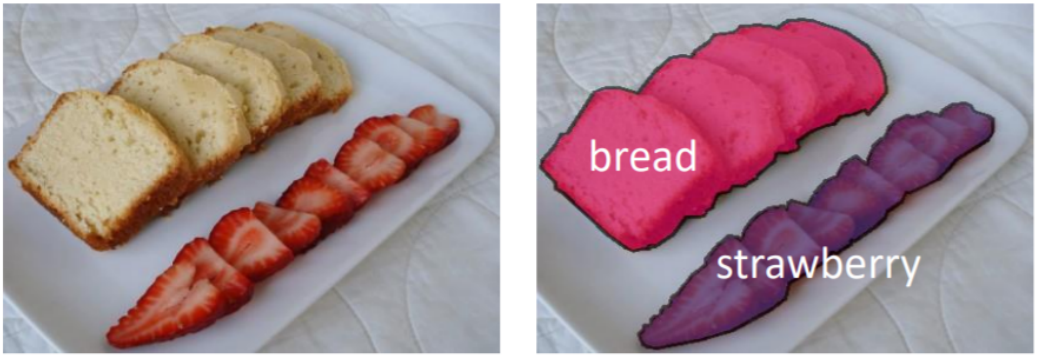


Figura 1.4: Exemplo de segmentação semântica aplicada ao contexto de detecção e classificação de alimentos [3].

A imagem apresentada na Figura 1.4 é de uma base de dados recente criada para testar e avaliar diferentes algoritmos de Segmentação Semântica. O *benchmark* FoodSeg103 [3] conta com 103 classes diferentes e mais de 9000 imagens anotadas para a tarefa de segmentação, ou seja, cada imagem original tem uma máscara (*ground truth*) correspondente em escala de cinza, sendo que cada alimento é representado por um nível de cinza diferente (*id* do alimento). Nesse caso em específico, o alimento *Bread* é representado pelo nível de cinza 58 e *Strawberry* pelo 30. Para facilitar a visualização, são utilizadas cores para mapear esses níveis *grayscale* em algo mais ilustrativo.

Nota-se uma evolução muito rápida dos algoritmos de segmentação semântica nos últimos anos. Um dos mais famosos e que produziu resultados espetaculares é UNET [10], uma rede neural profunda em formato *encoder decoder*: a primeira parte é responsável por codificar e extrair as características da imagem usando CNNs como RESNET [39], e a segunda parte responsável por reconstruir essa representação densa na imagem original, por meio de convoluções transpostas (ou deconvolução). Ao reconstruir a foto original, é possível atribuir uma classe a cada pixel, realizando finalmente a segmentação semântica. Sem esse processo granular de classificação, é impossível determinar as bordas do objeto com exatidão (ao contrário da regressão *bounding box* em detecção de objetos).

Outro algoritmo muito popular que ganhou força é o DeepLab [44], que tem como principal diferencial o uso de convoluções dilatadas (ou *atrous convolution*). Esse processo permite obter um ângulo de visão mais largo para uma mesma convolução, armazenando de forma efetiva características dos pixeis ao redor de determinada região. São particularmente úteis em arquiteturas de segmentação semântica, já que evitam mais camadas de convolução em redes naturalmente grandes e caras computacionalmente.

1.1.2.3 Extração

Grandes quantidades de dados compartilhados em vários sites permitem coletar informações relacionadas a alimentos, tais como receitas de texto, imagens e vídeos. Na área da saúde, a derivação do conteúdo nutricional a partir de imagens de alimentos requer o reconhecimento fino dos ingredientes. Entretanto, o reconhecimento direto dos ingredientes as vezes é um desafio,

uma vez que os ingredientes de alimentos preparados são misturados. Neste caso, pode-se recuperar receitas com base em uma consulta de imagem e uma breve descrição do alimento presente, caracterizando um modelo multi-modal.

Há inúmeras vantagens de saber a relação entre ingredientes e alimentos: por exemplo, a possibilidade de recomendar receitas beneficiará os usuários que querem cozinhar um prato em particular a partir de uma imagem disponível na internet. Além disso, a receita fornece informações ricas, tais como métodos de cozimento, ingredientes e suas quantidades, o que pode facilitar a estimativa da tabela nutricional. Um dos estudos mais relevantes nessa área é o IM2RECIPE [4], um sistema que recebe uma imagem de entrada e extrai a receita e as instruções para realizá-la. Uma rede neural profunda foi treinada com mais de 1 milhão de imagens para aprender uma representação conjunta de receitas e imagens e que produzem resultados impressionantes. A figura 1.5 mostra um exemplo:

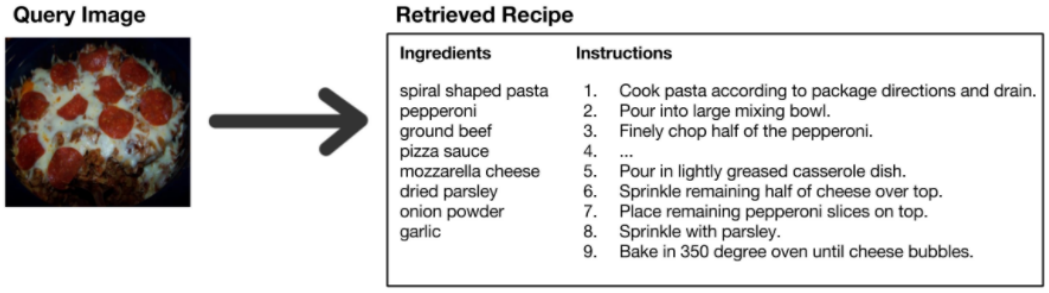


Figura 1.5: Exemplo de uso do sistema IM2RECIPE [4].

1.1.2.4 Recomendação

A recomendação de alimentos é um domínio importante tanto para os indivíduos quanto para a sociedade. Diferente de outros tipos de sistema de recomendação, a recomendação de alimentos envolve informações mais complexas, multifacetadas e dependentes do contexto (por exemplo, preferências de estilo de vida e cultura). A recomendação alimentar consiste em quatro tipos: métodos baseados no conteúdo, métodos baseados em filtragem colaborativa, baseados em algum determinado contexto e método voltado para a saúde.

Dentre as abordagens, recomendação de alimentos voltadas para a área da saúde costumam se destacar dado a relevância do problema e o crescente número de pessoas com maus hábitos alimentares. Por exemplo, [45] incorpora aspectos nutricionais na abordagem de recomendação com base na "função de equilíbrio calórico". Foi proposto um sistema de recomendação nutricional personalizado, que pode calcular com eficiência quais itens são os mais saudáveis e reordenam e filtram os resultados para os usuários com base em seus dados de saúde.

1.1.2.5 Predição e monitoramento

A predição e o monitoramento de mídias sociais podem fornecer várias informações relevantes da saúde populacional, possibilitando direcionar melhor certas decisões (*Big Data*). Por exemplo, o uso das mídias sociais para o monitoramento da saúde pública também tem sido cada vez

mais popular, principalmente quando tem Aprendizado de Máquina e Modelos Preditivos como aliados. Nos primeiros anos, estudos alimentares em larga escala sobre o consumo de alimentos utilizavam questionários e diários de alimentos para manter um registro de seus participantes, o que pode ser chato e trabalhoso de conduzir.

Alternativamente, as mídias sociais, tais como *Twitter* e *Instagram* oferecem a seus usuários uma forma de registrar suas vidas diárias, incluindo escolhas alimentares. Com uso de *Web Crawlers* e outras técnicas de Mineração de Dados, é possível compilar todas essas informações e processá-las para extrair *insights* que vão levar as melhores tomadas de decisões. Além disso, a rápida evolução de *Machine Learning* impulsionou as aplicações dessa tarefa, que é de extrema relevância para órgãos governamentais e empresas alimentícias na busca do entendimento dos hábitos alimentares em grande escala.

1.2 Descrição do problema e objetivos

Esse trabalho busca continuar um estudo feito por [5], em que buscou-se realizar a contagem automática de carboidratos a partir de uma foto e o peso total dos alimentos no prato. Essa contagem automatizada possibilita sugerir dose de insulina para diabéticos insulino-dependentes, agilizando um processo trabalhoso, chato e manual de cálculos matemáticos. Apesar do sucesso do estudo e inúmeros usuários teste, o aplicativo desenvolvido ainda é muito manual e não consegue reconhecer automaticamente os alimentos, sendo necessário que uma pessoa marque com o dedo as posições corretas e os tipos de alimentos na imagem (ver figura 1.6). Buscando formas de evoluir a ferramenta e deixá-la ainda mais rápida, uma alternativa natural é trabalhar com classificação e segmentação automática de alimentos. Além disso, implementar um sistema que automaticamente identifica os alimentos de uma imagem abre portas para inúmeras aplicações com foco na cultura brasileira.

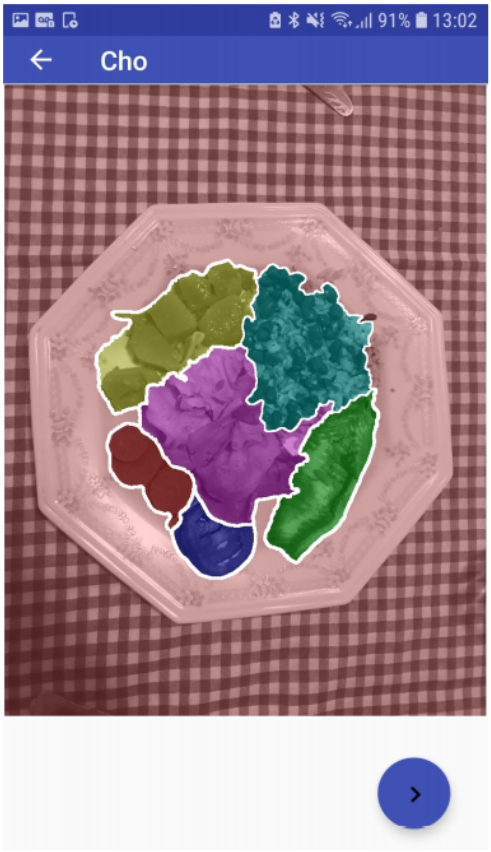


Figura 1.6: Marcações manuais do usuário ao usar o aplicativo [5].

Como já visto anteriormente na subseção 1.1.2.2, segmentação semântica busca classificar e segmentar objetos em uma imagem, sendo que em *Food Computing* esses objetos são os alimentos. Em comparação com sistemas que apenas classificam, a vantagem de segmentar os alimentos é conseguir extrair informações nutricionais em uma etapa posterior a partir da área de alimentos, atribuindo ainda mais valor à aplicação.

Podemos nos perguntar: por que não realizar a tarefa de detecção de objetos a fim de também extrair informações nutricionais. Apesar da detecção objetos (alimentos) em uma imagem dar informações de tamanho para os alimentos, estas não são precisas pois as bordas não são delineadas corretamente. A tarefa de detectar objetos atribui apenas um retângulo em volta do alimento indicando onde na imagem esses estão, porém não fornece exatidão nas fronteiras e na área de cada alimento. Por outro lado, segmentação semântica, apesar de ser uma tarefa mais complexa, consegue trazer mais confiabilidade e robustez na identificação da posição dos alimentos, já que é feita uma classificação pixel a pixel da imagem.

Apesar de inúmeras aplicações e estudos já realizarem segmentação semântica de alimentos, muitos deles com objetivo de auxiliar na gestão de dietas, nenhum desses trabalhos faz um estudo minucioso e comparativo de alimentos brasileiros. Sabe-se que a qualidade de sistemas de Inteligência Artificial é melhor e mais robusta quando estes são focados em determinado nicho. Hoje ainda não existe nenhum sistema de reconhecimento capaz de abranger todas as culturas

culinárias do planeta com exatidão, abrindo portas e oportunidades para que isso seja também desenvolvido no Brasil.

Tendo isso em vista, procurou-se base de dados de alimentos tipicamente brasileiros e anotados para a tarefa de segmentação semântica. A surpresa foi encontrar apenas um que satisfazia o propósito, sendo este criado em ambientes muito controlados, com normalmente um alimento por foto e poucas classes disponíveis. [21] conseguiu reunir 1250 imagens que tinham os seguintes alimentos: maçã, feijão, ovo cozido, frango, arroz, salada, carne, spaghetti, ovo frito e carne. Naturalmente, qualquer aplicação relevante para gestão de dietas ou avaliação de macronutrientes não consegue se desenvolver a partir de uma base tão reduzida de alimentos (pouca variedade). Por isso, foi necessário elaborar uma própria base de dados mais completa que realmente representasse o cotidiano de um brasileiro. A figura 1.7 mostra algumas imagens que foram coletadas e usadas neste projeto.

Figura 1.7: Imagens coletadas para criação da base de dados com alimentos brasileiros.

Não satisfeito com a falta de base de dados, não foi encontrada uma ferramenta ágil o suficiente para anotar o *Dataset* e sinalizar a posição e classe dos alimentos na foto. Com isso em mente, também foi desenvolvido um sistema simples que utiliza *Watershed* [46], uma técnica que separa regiões automaticamente a partir de marcações iniciais. Dessa forma, não é necessário delimitar de forma lenta e detalhada as fronteiras de cada objeto.

Com a base de dados criada, é necessário procurar na literatura modelos de aprendizagem profunda capazes de realizar essa tarefa, assim como avaliar sua performance no *Dataset* construído. A escolha de modelos *Deep Learning* é muito natural no dias atuais, tendo em vista que são os modelos supervisionados que dão os melhores resultados para problemas de visão computacional. Essas ideias abrem possibilidades para auxiliar a dieta de diabéticos e obesos, recomendar alimentos saudáveis, identificar padrões alimentares e estimar macronutrientes a partir de uma imagem. Definido o objetivo geral do trabalho com foco no público brasileiro, temos os seguintes objetivos específicos do projeto:

- Elaboração de uma base de dados relevante e que represente de forma realista os hábitos e cultura alimentar do brasileiro.
- Criação de uma ferramenta para anotar as imagens da base criada, com o objetivo de sinalizar quais alimentos estão na foto e em qual posição (fundo verdade ou *ground truth*).
- Implementação de um algoritmo de segmentação semântica usando os modelos e arquiteturas mais recentes de aprendizagem profunda.
- Elaboração dos cenários de teste do nosso algoritmo, incluindo testes com bases públicas e com a própria base criada. Além disso, serão testadas diferentes agrupamentos de classes para a base criada com o intuito de observar o efeito dos resultados ao aumentar ou diminuir a granularidade das classes (e consequentemente a complexidade da tarefa).
- Avaliação dos resultados e comparação das métricas com outros estudos similares.

A Figura 1.8 resume bem o que foi feito nesse trabalho.

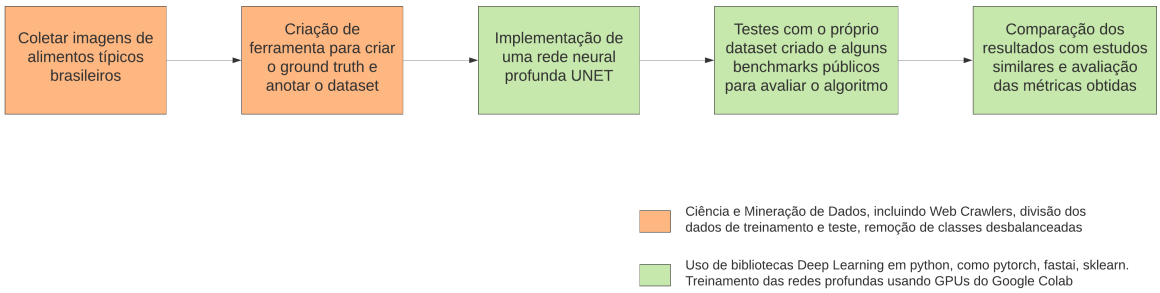


Figura 1.8: Fluxograma resumindo os objetivos específicos do trabalho.

1.3 Apresentação do manuscrito

Este manuscrito consiste dos seguintes capítulos:

- Fundamentos teóricos (Capítulo 2), responsável por detalhar e dar um embasamento científico para todas as tecnologias utilizadas nesse trabalho. Serão descritos termos gerais, como inteligência artificial, *Deep Learning*, redes neurais, assim como termos específicos do trabalho, como por exemplo, modelos de arquitetura RESNET e UNET, técnicas de otimização ADAM e *Fit-one-cycle*, funções de custo *Dice Loss*, *Focal Loss* e *Cross Entropy Loss*.
- Trabalhos relacionados (Capítulo 3, capítulo que descreve outros estudos similares à este. Como o foco é reconhecimento e segmentação de alimentos, serão mostrados outros trabalhos que fizeram isso e seus respectivos resultados. Será apresentado também as tecnologias que esses utilizaram e uma comparação entre diferentes arquiteturas. Por último, é feito um resumo de todas as bases de dados de alimentos públicas disponíveis para uso e que servirão para testar nossos modelos implementados.

- Metodologia (Capítulo 4) detalha de forma minuciosa todas as etapas implementadas nesse trabalho. Primeiro mostra-se como foi feita a elaboração da base de dados do projeto com alimentos brasileiros, desde a coleta dos dados crus até a anotação das imagens para realizar o treinamento supervisionado. Também fala-se do algoritmo de segmentação utilizado e os parâmetros escolhidos para realizar os testes. Finalmente, são descritos os cenários de teste e as métricas de avaliação dos modelos e bases testadas.
- Resultados (Capítulo 5) apresenta concisamente tudo que foi atingido, seguindo os cenários de teste e métricas já apresentadas na metodologia. São mostrados os gráficos de treinamento, tabelas comparativas, métricas de avaliação e análise dos resultados para diferentes modelos e bases de dados. Também é discutido a qualidade dos resultados e comparado com outras tarefas de Visão Computacional, como reconhecimento de outros objetos e cenas.
- Conclusão (Capítulo 6) finaliza este estudo resumindo tudo que foi realizado, como os principais avanços feitos e o melhores resultados atingidos. Busca-se também mostrar perspectivas futuras, mostrando diferentes caminhos que podem ser seguidos para melhorar os modelos e arquiteturas implementadas e a base de dados criada.

Capítulo 2

Fundamentos Teóricos

Neste capítulo serão apresentados os principais conceitos teóricos necessários, estes relativos ao desenvolvimento das atividades propostas neste trabalho.

2.1 Conceitos relacionados à construção da base de imagens

2.1.1 DHASH

O algoritmo *Difference Hashing (DHASH)* [47] foi utilizado para remover imagens duplicadas da base de imagens. Funções digestoras ou *hash* são funções matemáticas capazes de computar um resumo de determinado conteúdo digital. Tais funções tem como objetivo gerar uma saída de tamanho fixo a partir de uma entrada de tamanho variável. Além disso, deseja-se construir funções digestoras que dado uma mínima mudança no conteúdo de entrada, o valor computado de saída mude também, ou seja, que conteúdos diferentes não produzam a mesma saída da função de *hash*.

Como deseja-se criar um resumo de alguns bits que represente uma imagem, há alguns passos para redução de informações menos necessárias. O primeiro passo consiste em converter a imagem para 9x8, totalizando 72 pixels. Após esse passo, transforma-se a imagem em tom de cinza. Por se tratar de um algoritmo que foca no gradiente da imagem, compara-se bits vizinhos para atribuir valores, ou seja, se o pixel analisado é menos brilhante que o pixel a sua esquerda, recebe o valor 0, caso contrário, recebe o valor 1. Com 9 colunas, são realizadas 8 comparações em 8 linhas, resultando em um resumo de 64 bits.

2.1.2 Algoritmo *WaterShed*

O algoritmo *WaterShed* [46] é utilizado principalmente para realizar segmentação de áreas que se tocam. Isto é, a partir de traços ou pontos realizados pelo usuário que identifiquem as regiões a serem segmentadas, o algoritmo interpreta tais entradas como elevações locais. Interpretar a imagem como uma "topografia"permite que, a partir das elevações locais, compute-se um derramamento (crescimento da área segmentada) até que toque outra área.

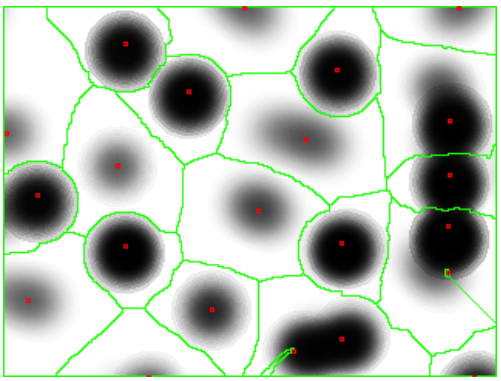


Figura 2.1: Exemplo de saída de algoritmo de Watershed [6].

Em termos mais matemáticos, interpreta-se o gradiente da imagem como uma superfície topográfica onde os mínimos são os pixels que representam o plano de fundo e as entradas do usuário são vistos como elevações nessa superfície.

2.1.3 Data Augmentation

Uma técnica bastante comum ao se tratar de treinamento de redes é o aumento de dados ou *Data Augmentation*. Essa técnica consiste em aplicar mudanças (filtros, rotações, recortes) as imagens da base de dados para conseguir aumentar a quantidade de imagens na base original. Inicialmente a técnica era utilizada para gerar uma base nova, a partir da base original, salvando os novos arquivos e utilizando-os posteriormente. Novas biblioteca/*frameworks* como o [7] realiza o aumento de dados *"on-the-fly"*, ou seja, os *batches* são criados a medida em que são necessários pelo treinamento e tais mudanças são aplicadas a cada *batch* de amostras. Esse procedimento evita que a base de dados aumentada fique em armazenamento local e permite ser gerada a medida que as imagens são necessárias (Figura 2.2).

(a) Crop (b) *Rotação*

Figura 2.2: Exemplos de Data Augmentation [7].

Um ponto importante a ser ressaltado é que há uma linha tênue na utilização da técnica. Por um lado, aumenta-se a diversidade de dados, reduz-se o *bias* e aumenta a capacidade de generalização do modelo. Porém, de outro lado está a capacidade de aprendizagem do modelo e a repetição de dados. A medida que forem aplicadas técnicas e as imagens modificadas forem utilizadas para treinar a rede, a variação entre cada nova imagem gerada agrega pouco valor para o treinamento, já que a rede já "viu"algo similar. Portanto, não deve-se exagerar, evitando um

cenário em que há imagens muito similares entre si na base e conduzindo o treinamento para um caso de *overfitting*.

2.2 Conceitos de redes neurais relevantes para segmentação semântica

2.2.1 Redes neurais convolucionais

Redes neurais convolucionais [8] são redes de neurônios artificiais bio-inspirados na organização do córtex visual. Assim como a rede neural humana, as redes convolucionais tem a capacidade de aprender quais são as características importantes para classificar uma imagem. Geralmente, as redes convolucionais são utilizadas em conjunto com redes *Fully Connected* especialistas em classificação, assim uma parte da rede tem a função de extrair e aprender características importantes da imagem (*feature learning*) e outra parte tem a função de classificar as características extraídas.

O procedimento de extração de informação importante é feito em cascata. A partir de uma imagem inicial, extrai-se as principais características e logo em seguida aplica-se funções que resumam tal imagem (reduz-se dimensões) gerando um mapa de informações importantes (*feature map*) que será utilizada como entrada para outra camada convolucional. Assim, reduz-se a informação iterativamente até que as dimensões se reduzam a um vetor. Esse vetor define as características mais importantes da imagem e é utilizado para classificação.

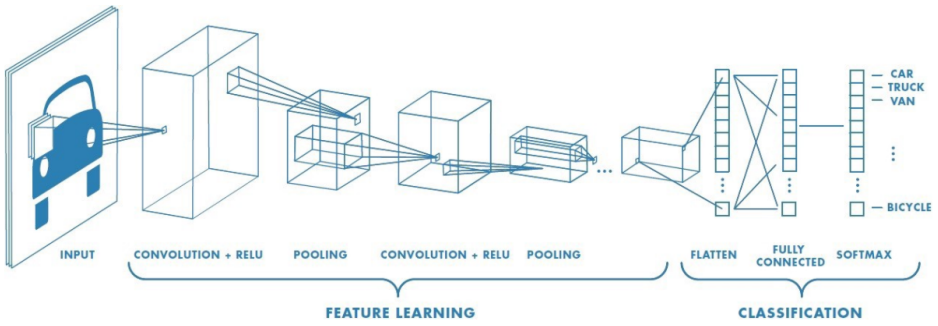


Figura 2.3: Redes Neurais Convolucionais [8].

Uma prática comumente usada é importar pesos para a parte de convolução. Isto é, utiliza-se pesos de redes já treinadas e que obtiveram resultado significativo na extração de características importantes. Por exemplo, muitas vezes se utiliza pesos de redes treinadas com o banco de imagens (IMAGENET) [38] para a parte convolucional e foca-se no processo de classificação tendo as *features* como garantidas. Esse procedimento é conhecido também como *transfer learning*.

2.2.2 Resnet

Resnet, abreviação de *Residual Network*, é um tipo específico de rede neural que foi introduzida em 2015 no trabalho "*Deep Residual Learning for Image Recognition*"[39].

A fim de resolver um problema complexo, empilha-se algumas camadas adicionais nas Redes Neurais Profundas, o que resulta em maior precisão e desempenho. A intuição por trás da adição

de mais camadas é que estas camadas aprendem progressivamente características mais complexas. Por exemplo, em caso de reconhecimento de imagens, a primeira camada pode aprender a detectar bordas, a segunda camada pode aprender a identificar texturas e, da mesma forma, a terceira camada pode aprender a detectar objetos e assim por diante. Mas descobriu-se que existe um limite máximo de profundidade para o modelo tradicional de rede neural convolucional, onde a partir desse limite, o desempenho da rede começa a deteriorar.

Este problema de formação de redes muito profundas foi aliviado com a introdução da Resnet, formada a partir de Blocos Residuais. A principal diferença é que existe uma conexão direta que salta algumas camadas, criando um elo direto entre estas. Esta conexão é chamada de *skip connection* e é o núcleo dos Blocos Residuais, conforme mostra a figura 2.4.

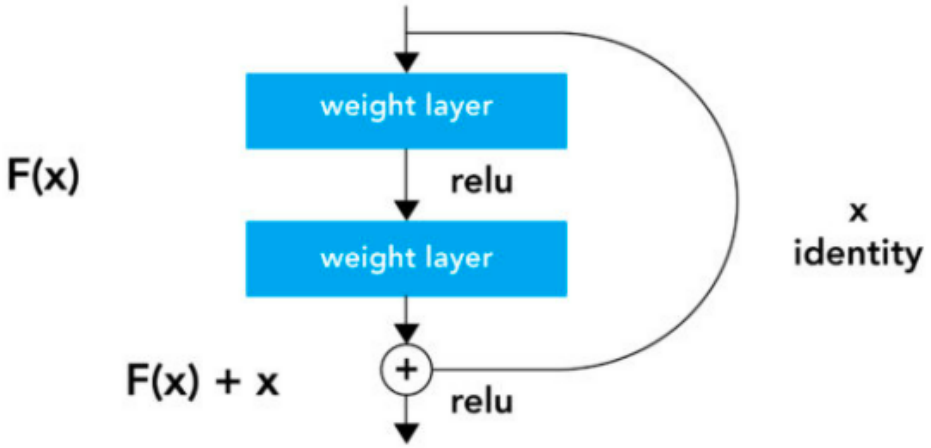


Figura 2.4: Bloco Residual da arquitetura Resnet [9].

As *skip connections* na Resnet resolvem o problema de *vanishing gradients* em redes neurais profundas, permitindo que este caminho de atalho alternativo para o gradiente flua através dele. A outra maneira que estas conexões ajudam é permitindo que o modelo aprenda as funções de identidade, o que garante que a camada superior tenha um desempenho pelo menos tão bom quanto a camada inferior, e não pior.

Além disso, a arquitetura Resnet é caracterizada como espinha dorsal *backbone* de uma rede neural convolucional, sendo responsável por codificar informações em um espaço vetorial reduzido. Essa possui variações de tamanho, sendo Resnet18 uma rede residual com 18 camadas de parâmetros, Resnet34 com 34 camadas e assim por diante. Quanto maior o tamanho da Resnet, maior a capacidade de aprendizado, porém é necessário mais dados de treinamento para que seja notada essa diferença de aprendizado.

2.2.3 Unet

Unet [10] é uma arquitetura de rede neural convolucional especializada na segmentação de imagens. As redes convolucionais geralmente reduzem a informação a um vetor. Para problemas de classificação, isto é suficiente. Porém quando se trata de segmentar a imagem, deseja-se obter uma imagem (com a segmentação de cada classe) a partir do vetor de características extraídas. Esse procedimento de expansão utiliza das camadas de convolução para realizar o caminho contrário. Por se tratar de uma arquitetura tanto de compressão de informação quanto de expansão (*Auto-Encoder*), ela é nomeada de Unet, ou rede em forma de U.

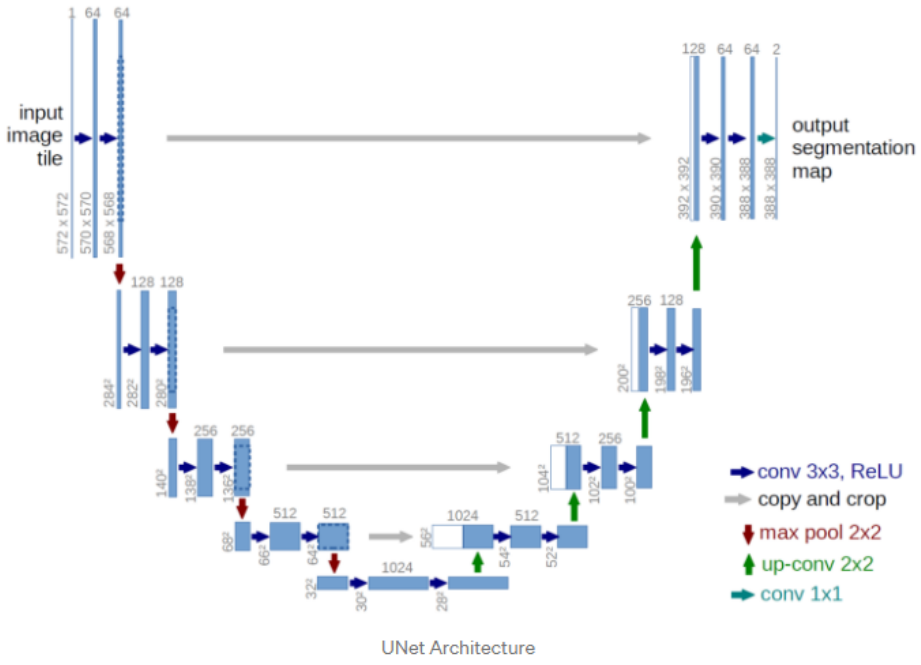


Figura 2.5: Perfil da arquitetura da UNet [10].

2.3 Estratégias de Otimização

Para um problema de aprendizado profundo, geralmente define-se primeiro uma função de perda. Uma vez que se tem a função de perda, podemos usar um algoritmo de otimização na tentativa de minimizar a perda. Na otimização, uma função de perda é frequentemente referida como a função objetiva do problema de otimização. Por tradição e convenção, a maioria dos algoritmos de otimização se preocupa com a minimização. E a escolha do algoritmo de otimização para seu modelo de aprendizado profundo pode significar a diferença entre bons resultados em minutos, horas e dias.

2.3.1 Funções de *Loss*

Em termos gerais, funções de custo (ou perda) são funções que avaliam quão bem um modelo está aprendendo determinada tarefa e são usadas para otimizar as predições da rede neural. Se

as predições do modelo estão bem próximos do desejado, a função de *loss* ou custo retornará um valor baixo, caso contrário retornará um valor alto, ou seja, o erro ainda está grande. Um algoritmo de otimização, como Gradiente Descendente, tentará reduzir o erro dessa função de custo alterando os parâmetros do modelo. Há algumas maneiras diferentes de computar tais funções e duas delas serão descritas a seguir.

2.3.1.1 Cross Entropy Loss

A entropia cruzada [48] é uma medida do campo da teoria da informação, baseada na entropia e geralmente calcula a diferença entre duas distribuições de probabilidade para uma dada variável aleatória ou conjunto de eventos. É uma das funções de custo mais importantes, utilizada para otimizar modelos de classificação.

Antes de entender sua utilidade no contexto de função de custo, primeiro é preciso entender o que significa entropia. A figura 2.6 traduz bem o conceito, mostrando o valor de entropia em função de diferentes probabilidades de uma variável *booleana* ser verdadeira ou falsa. Quando p = 0, essa variável com certeza é falsa. Quando p = 1, essa variável com certeza tem valor verdadeiro. Porém, entre esses dois extremos, a probabilidade da variável *booleana* ser verdadeira ou falsa é incerta, portanto a entropia aumenta representando o grau de incerteza do sistema. O valor máximo de entropia naturalmente acontece quando p = 0.5, onde o sistema não consegue decidir o valor da variável.

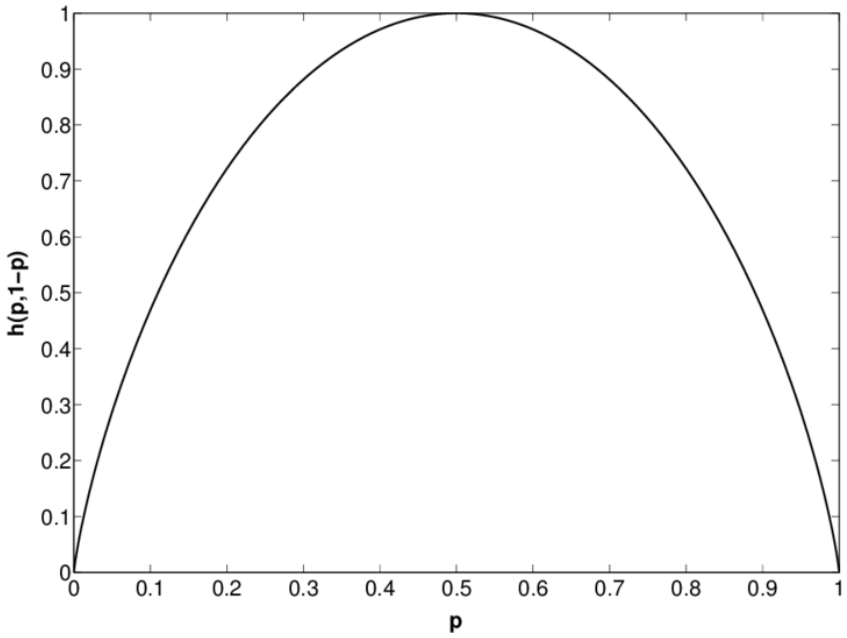


Figura 2.6: Gráfico da Entropia H em função de uma probabilidade p [11].

Traduzindo a explicação para uma situação mais cotidiana, basta imaginar que essa variável *booleana* representa a classificação de determinado objeto na cena. Por exemplo, uma rede neural tenta identificar que há um gato na imagem: se a saída da rede for p = 0, ela está bem confiante

que não há gatos. Porém se p = 1, há grandes chances de ter gatos. Mas se ela estiver confusa, p = 0.5 é um valor esperado.

Essa função pode ser usada como função de custo, pois o objetivo de otimizar uma rede neural é justamente fazer que a predição seja próxima do verdade de referência (ou rótulos). Toda vez que a saída da rede for indefinida (p = 0.5), a entropia é alta, portante a função de custo aumenta. Se a rede tiver aprendido determinada tarefa, a entropia será baixa, pois estará confiante da predição.

Apesar de intuitiva a explicação sobre entropia, ainda faltam detalhes na definição de Entropia Cruzada. Como dito anteriormente, ela calcula a diferença entre duas distribuições de probabilidade. É esperado que as probabilidades das predições da rede sejam otimizadas para serem próximas da distribuição do fundo verdade (rótulos). É necessário, portanto, cruzar essas distribuições e obter uma equação que penalize diferenças entre elas, de acordo com:

$$L_{crossentropy}(y, y_{hat}) = -\sum_{i} y_i log(y_{hat_i}) \tag{2.1}$$

A fórmula 2.1 mostra esse cruzamento de distribuições. y é a saída esperada da rede (rótulo) e y_hat representa a predição da rede neural. Se determinada classe i está presente na imagem (y=1), o objetivo é que o modelo seja otimizado para produzir uma saída y_hat = 1 também. Se isso acontecer, a função de custo terá sido a mínima possível, já que log(1) = 0. Mas se a rede neural ainda não produzir esse resultado, a função de custo estará elevada, com log(y_hat) diferente de zero. Por fim, entende-se que essa função de custo é ótima para modelos de classificação (saída discreta), sendo uma ferramenta importante para treinar modelos de inteligência artificial.

2.3.1.2 Focal Loss

A função de custo *Focal Loss* [12] reduz o desbalanceamento de classes durante o treinamento em tarefas como a detecção de objetos. A *Focal Loss* aplica um termo modulador à equação de entropia cruzada a fim de focar o aprendizado em amostras difíceis (é possível detectar essas amostras a partir da baixa confiança do modelo ao classificar determinada amostra, por exemplo). É uma função de entropia cruzada em escala dinâmica, onde o fator de escala decai para zero à medida que aumenta a confiança da classe correta. Intuitivamente, este fator pode automaticamente diminuir a contribuição de exemplos fáceis durante o treinamento e rapidamente focalizar o modelo em exemplos difíceis.

Formalmente, a perda focal acrescenta um fator ao critério padrão de entropia cruzada. O ajuste *gamma*, quando maior que zero, reduz a perda relativa para exemplos bem classificados, colocando mais foco em exemplos difíceis e mal classificados. Essa ideia pode ser vista na figura 2.7. Note que quanto maior *gamma* (curva verde), menor é a contribuição dos exemplos bem classificados para a função de custo. Já a curva azul representa *gamma* igual a zero, ou seja, a própria função de entropia cruzada, onde não há preocupação em atribuir pesos diferentes para as amostras durante a otimização dos parâmetros da rede.

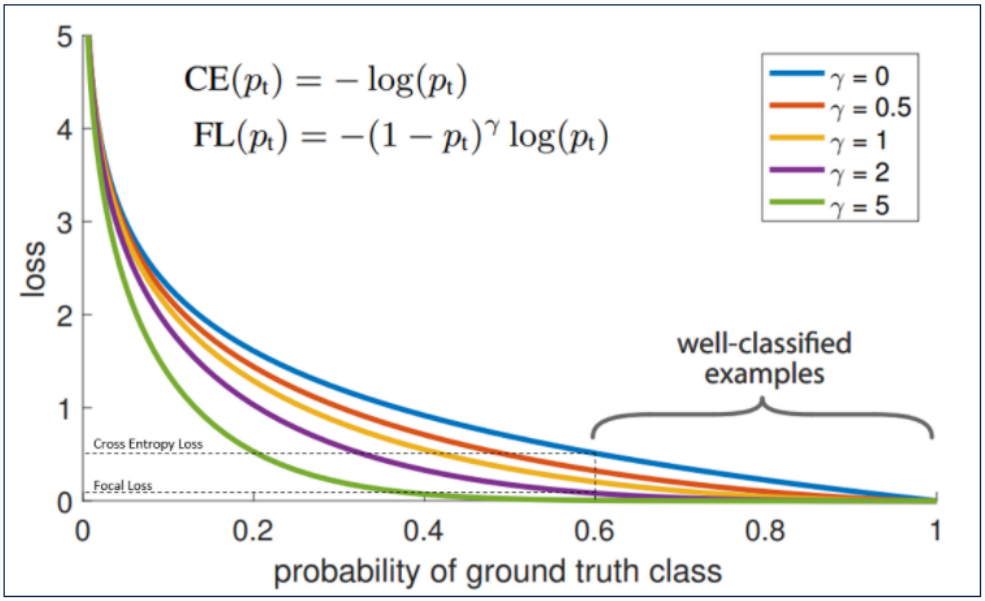


Figura 2.7: Comparação da *Focal Loss* com *Cross-Entropy Loss*. Observa-se também as fórmulas de ambas funções e diferentes curvas de perda dependendo do valor de *gamma* [12].

2.3.2 Adam

O algoritmo de otimização Adam [13] é uma extensão do gradiente descendente estocástico e que recentemente é usado como otimizador padrão [7] em aplicações de aprendizagem profunda em visão computacional e processamento de linguagem natural.

Gradiente descendente estocástico mantém um único *learning rate* (denominado *alpha*) para todas as atualizações de peso e essa taxa de aprendizagem (*learning rate*) não muda durante o treinamento. Em Adam, a taxa de aprendizagem é mantida fixa para cada peso de rede (parâmetro) e adaptada separadamente à medida que a aprendizagem se desdobra, dependendo de como o gradiente está variando.

Em vez de adaptar a taxa de aprendizagem dos parâmetros com base na média do primeiro momento (a média), Adam também faz uso da média dos segundos momentos dos gradientes. Especificamente, o algoritmo calcula uma média móvel exponencial do gradiente e do gradiente quadrado, e os parâmetros beta1 e beta2 controlam as taxas de decaimento dessas médias móveis. Isso permite encontrar o mínimo global da função de custo de forma muito mais rápida e evita a explosão dos gradientes. A figura 2.8 mostra um comparativo de vários algoritmos de otimização e revela que Adam encontra, em geral, uma função de custo menor e de forma mais rápida que outros algoritmos.

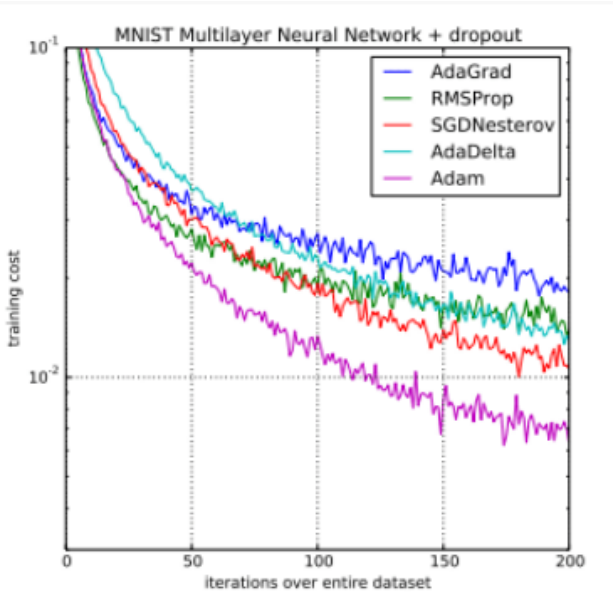


Figura 2.8: Comparação da função de custo de treinamento para vários algoritmos de otimização [13].

2.3.3 *Fit one Cycle*

No treinamento de uma rede neural profunda, normalmente combina-se o algoritmo de otimização com um "planejador"(ou *scheduler*), responsável por variar o *learning rate* ou o momento do algoritmo de otimização. Aqui pode-se criar confusão, pois Adam por exemplo, já varia a taxa de aprendizagem ao longo do treinamento. Na verdade, Adam apenas muda um fator de escala que multiplica *alpha*, dependendo da variação dos gradientes em cada parâmetro da rede. Já um *scheduler* muda ativamente o parâmetro *alpha*, contribuindo para deixar o algoritmo de otimização final ainda mais dinâmico.

Uma dessas estratégias de variar e planejar hiperparâmetros de otimização é chamada de *Fit one cycle* [49], onde tanto o *learning rate* quanto o momento variam de forma oposta, com o objetivo de facilitar a busca pelo mínimo global da função de custo. A estratégia também é chamada de superconvergência, pois converge rapidamente os parâmetros da rede neural. A figura 2.9 mostra como esses dois hiperparâmetros variam ao longo das épocas de treinamento, prevenindo *overfitting* e potencializando a minimização da função de custo (*loss*).

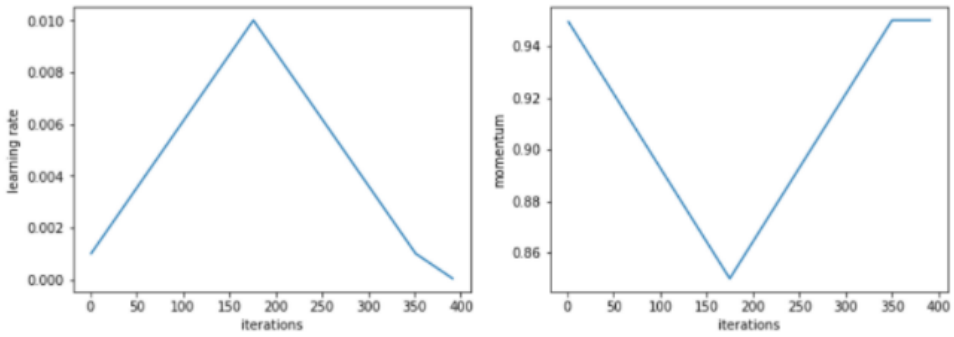


Figura 2.9: Variação do *Learning Rate* e do Momento durante as épocas de treinamento [7].

2.4 Estratégias de Avaliação

2.4.1 *Cross Validation*

Cross Validation [14] é um método estatístico utilizado para reduzir viés no treinamento de redes neurais. Muitas vezes divide-se os dados em conjuntos de treinamento e conjunto de teste, porém essa divisão tem um problema. Ao se dividir, por exemplo, uma base de dados em 60% para treinamento e 40% para teste, pode haver um viés nessa separação. Ou seja, muitas amostras de uma mesma classe podem estar no conjunto de treinamento e poucas no de teste, resultando em uma baixa taxa de acerto.

A técnica de validação cruzada consiste em subdivir toda a base de dados em pastas. Ou seja, para um K igual a 5, será subdividido em cinco grupos. Após essa divisão, varia-se quais as pastas serão utilizadas para treinamento e quais as pastas que serão utilizadas para teste. Realiza-se repetidas iterações mantendo a proporção treinamento/teste, porém variando quais pastas ou *folds* estarão em cada grupo.

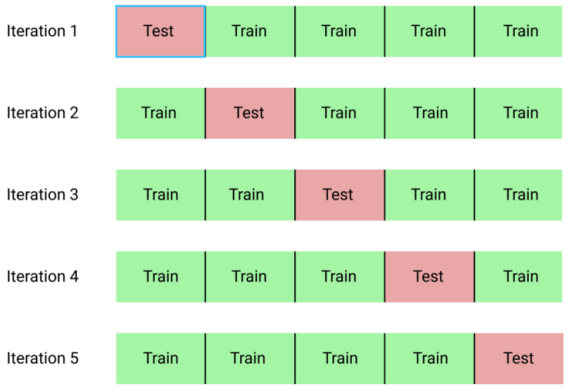


Figura 2.10: Técnica estatística - KFold - para reduzir viés no treinamento [14].

2.4.2 IoU - *Intersection Over Union*

A métrica IoU ou *Intersection Over Union* [15] é uma das formas de avaliar os resultados apresentados por uma rede de segmentação. Se trata de uma métrica que compara a área predita/segmentada pela rede com a área definida como verdade pelo *ground-truth*. Realiza-se então o cálculo do número de pixels da intersecção entre as duas áreas e o cálculo do número de pixels da união das duas áreas. A métrica (IoU) é dada pela razão entre a intersecção e a união, variando de 0 a 1, em que 0 representa segmentação sem nenhuma relação com a anotação e 1 representa correlação perfeita entre a predição e o *ground-truth*.

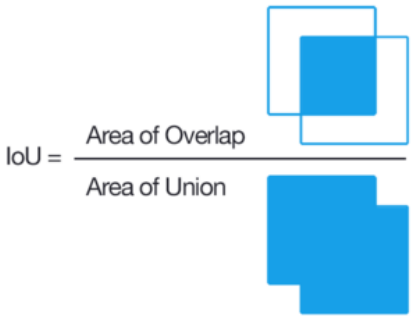


Figura 2.11: Intersecção sobre União [15].

Associado a esta métrica, há um limite que define casos de erro e casos acerto. Inicialmente, o limite de IoU igual a 50% era considerado um caso de acerto, ou seja, a rede conseguiu segmentar suficientemente bem a área. Porém, com o avanço de redes segmentadoras, considera-se atualmente que um limite de 75% seja mais adequado e mais criterioso ao decidir se a rede acertou ou não a tarefa de segmentar determinada área.

Em casos de segmentação de múltiplas áreas em uma mesma imagem, utiliza-se a métria mIoU, que realiza uma média do valor de IoU para todas as classes.

2.4.3 F1 - Score

Uma outra métrica utilizada para avaliar processos de segmentação é F1-Score [15]. Essa métrica difere da métrica anterior por calcular duas vezes a intersecção sobre a soma da quantidade de pixels em ambas as imagens.

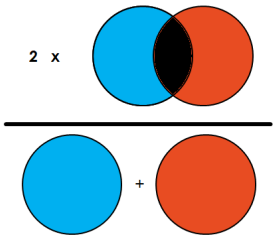


Figura 2.12: F1 Score - 2 x interseção sobre total de ambas as áreas [15].

Ambas as métricas podem ser descritas em termos de Verdadeiros Positivos, Falsos positivos, Verdadeiros Negativos e Falsos negativos. A partir da matriz de confusão, é possível extrair os valores de precisão e revocação (*precision and recall*) que se correlacionam com as métricas apresentadas da seguinte forma.

$$IoU = \frac{Precision * Recall}{Precision + Recall - Precision * Recall} \tag{2.2}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.3}$$

Em que Precisão ou *Precision* ilustra estatisticamente o percentual de identificações verdadeiras que realmente estavam corretas.

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

Já *Recall* representa o percentual de verdadeiros positivos foi identificado corretamente.

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

2.4.4 Acurácia para Segmentação Semântica

A acurácia é uma métrica para avaliar modelos de classificação. Informalmente, ela é a fração das predições corretas do modelo. Formalmente, ela tem a seguinte definição 2.6:

$$Accuracy = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{2.6}$$

Apesar de segmentação semântica ser uma tarefa de classificação, esta tem suas peculiaridades. Normalmente é feita a classificação de cada pixel na imagem, portanto a acurácia global será a média dos acertos na classificação de cada um desses pixeis. Naturalmente, nota-se um problema nessa análise: existem muito mais pixeis que representam o fundo da imagem ao invés dos objetos em si. Se o modelo acertar grande parte do fundo, a acurácia global será elevada, mesmo que a classificação dos pixeis do objeto estejam erradas.

Essa não é uma métrica desejada de avaliação, pois não traduz bem a qualidade de predição do modelo. Uma forma de mitigar esse problema é retirar da análise todos os pixeis de fundo e calcular a média da acurácia apenas para os pixeis dos objetos detectados. Mesmo assim, os resultados podem ser enganosos: quando a representação de determinada classe é pequena dentro da imagem, a métrica será tendenciosa e atribuirá maior peso às classes de objetos maiores (mais pixeis presentes). É preferível usar IoU como métrica principal ao avaliar modelos de segmentação semântica, mas a acurácia que desconsidera pixeis de fundo pode revelar o desempenho global do modelo.

Capítulo 3

Trabalhos Relacionados

Dada a contextualização do problema, é importante ver os principais avanços na área de *Food Computing* para estudar as abordagens de outras aplicações semelhantes. *Food Computing* é uma subárea de conhecimento, que utiliza métodos da ciência da computação para realizar estudos relacionados a alimento. Os estudos nessa área envolvem a aquisição e análise de dados de alimentos, utilizando-se de recursos como visão computacional, aprendizado de máquinas, mineração de dados e outros adventos tecnológicos com o intuito de prover serviços para melhorar a saúde humana, guiando o comportamento humano e entendendo a cultura relacionada.

Diversas aplicações recentes, como a recomendação de alimentos saudáveis para melhorar o hábito alimentar de pessoas [25], a classificação de imagens de alimentos utilizando aprendizado profundo [26], estimação de carboidratos para auxiliar na contagem de carboidrato [27], estimação de calorias a partir de imagens de alimentos [28] e a recomendação de receitas para indicar alimentos a partir dos ingredientes na foto [4] mostram a relevância e as diferentes possíveis abordagens nessa área.

Para melhor aprofundamento nos trabalhos relacionadas a essa disciplina, o *Survey* [1] apresenta com detalhes técnicos diversos artigos publicados nos últimos anos e faz uma comparação de seus resultados para diversas aplicações. Além disso, consegue fazer um excelente resumo de *Datasets* públicos disponíveis contendo imagens de alimentos anotadas para tarefas específicas, como reconhecimento de alimentos, segmentação semântica e até predição de receitas alimentícias. Considerando o foco do trabalho atual, fica claro a necessidade de aprofundar nos estudos que buscam classificar alimentos, segmentá-los e extrair informações relevantes, como peso, volume e até nutrientes. Tendo isso em vista, pode-se dividir os trabalhos relacionados em 3 grandes subseções:

- Reconhecimento de alimentos
- Segmentação semântica de alimentos
- Estimativa de calorias e valores nutricionais

Em cada uma delas será mencionado os estudos relevantes mais recentes, assim como base de

dados disponíveis para auxiliar este trabalho. Por último, também será feita uma análise de aplicativos de celulares que estão hoje no mercado e se encaixam dentro das áreas de interesse já mencionadas. É interessante aprofundar neste último tópico, pois sabe-se que essa é uma área em constante evolução. Por exemplo, recentemente diversos modelos de celulares vêm equipados com múltiplas câmeras e APIs de Realidade Aumentada, o que possibilita inovar em aplicações de *Food Computing*.

3.0.1 Reconhecimento de alimentos

Reconhecer quais os alimentos estão presentes numa foto costuma ser o primeiro passo de qualquer aplicação nessa área, e, portanto, é a que apresenta um maior número de artigos e trabalhos relacionados. Sem a informação da classe dos alimentos, não é possível extrair nenhum dado relevante para posterior análise.

É possível dividir essa tarefa em 2 grandes áreas: classificação de imagens *Single-label* e classificação *Multi-label*. A primeira consiste em atribuir apenas uma classe de alimento à foto, ou seja, normalmente essa imagem apresenta apenas um alimento. Já a segunda não tem limite de uma classe por foto, sendo mais desafiadora pela necessidade de compreender múltiplos alimentos.

A seguir, será feita uma análise da principais bases de dados públicas para tal tarefa, assim como os estudos que as usaram.

3.0.1.1 Datasets

Em [1] é feito um bom resumo dos principais *Datasets*, conforme pode ser observado na tabela 3.1.

Dentre tantas opções, há algumas que se destacam pela variedade de classes de alimentos e também pela quantidade total de imagens. Por exemplo, Food-101 [37] contém 101 tipos de alimentos diferentes com 1000 imagens cada, sendo estes predominantemente ocidentais, incluindo bastante *fastfood*. Outra opção também mais antiga e popular pelo tamanho e quantidade de classes é o UECFOOD256 [50], porém este com foco maior na cultura oriental, especialmente no Japão. Esta base de dados, além de conter a classes do alimentos na foto, apresenta um diferencial em relação ao anterior, pois também tem anotado a posição de cada alimento no prato, permitindo realizar tarefas de detecção de objetos, além do reconhecimento.

Mais recentemente, [51] fez um trabalho interessante ao juntar vários desses *Datasets* em um único lugar. Possuindo 524 classes de alimentos para reconhecimento, Food524DB reúne VIREO, Food-101, Food50 e uma versão modificada de UECFOOD256. Portanto, consegue reunir uma gama maior de culturas e assim generalizar melhor a tarefa de reconhecimento de alimentos. No ano seguinte à publicação desse trabalho, os autores refinaram o trabalho e publicaram o FOOD475 [52], no qual classes redundantes foram removidas.

Por último e não menos importante, ISIA-FOOD500 [23] reúne a melhor e mais completa base de imagens para essa tarefa. Ele conseguiu aumentar o volume de fotos para aproximadamente 400.000 imagens, com as 500 principais classes de alimento segundo o Wikipedia. Em cada classe,

Tabela 3.1: Principais *Datasets* de alimentos [1].

Nome do *dataset*	**Número de imagens**	**Número de classes**	**Tarefa**
PFID	4545	101	Reconhecimento
Food50	5000	50	Reconhecimento
Food85	8500	85	Reconhecimento
UEC Food100	14361	100	Reconhecimento
UEC Food256	25088	256	Reconhecimento
ETHZ Food-101	101000	101	Reconhecimento
UPMC Food-101	90840	101	Reconhecimento
UNICT-FD889	3583	889	Extração
FooDD	3000	23	Detecção
Food201-Segmented	12625	201	Segmentação semântica
UNIMIB15	2000	15	Reconhecimento
UNIMIB16	1027	73	Segmentação semântica
Food-975	38785	975	Reconhecimento
Food500	148408	508	Reconhecimento
Instagram800K	808964	43	Reconhecimento
Food11	5000	50	Reconhecimento
UNICT-FD1200	4754	1200	Reconhecimento e extração
ECUSTFD	2978	19	Estimação de calorias
Food524DB	247636	524	Reconhecimento
Instagram 1.7M	1.7M	101	Análise e padrão de consumo

os pesquisadores fizeram questão de usar pelo menos 500 imagens por classe e conseguiram uma variedade de cultura alimentícia como em nenhum *Dataset*, incluindo as principais culturas ocidentais e orientais.

3.0.1.2 Estudos recentes

Dada a grande variedade de dados nessa tarefa de *Food Computing*, é esperado que também existam diversos estudos para melhorar a taxa de acerto no reconhecimento de alimentos. [1] faz novamente um bom compilado de vários estudos, o qual foram se desenvolvendo ao longo dos anos.

O artigo com maior relevância é o [53], que apresenta os melhores resultados para três grandes bases de dados. Inspirado pelo recente sucesso de rede profunda residual, os pesquisadores propõe um esquema de aprendizagem chamada *Wiser* e introduzem um bloco de convolução em fatias para capturar as camadas verticais dos alimentos. As saídas dos blocos residuais profundos são combinado com a convolução fatiada para produzir a pontuação de classificação para categorias específicas de alimentos. Em outras palavras, os autores modificaram Redes Neurais Residuais CNN e introduziram outro tipo de convolução apenas na direção do eixo vertical. Isso foi feito, pois acredita-se que realizar a convolução apenas nessa eixo pode extrair informações relevante de alimentos com várias camadas, como por exemplo uma torta ou uma lasanha.

Além desse trabalho, os próprios criadores do ISIA-FOOD500 [23] desenvolveram uma arquitetura especial de Redes Neurais CNN para resolver o mesmo problema de reconhecimento. Utilizando de artifícios como *Attention Mechanisms*. Os pesquisadores propuseram uma rede de atenção global-local empilhada chamada *SGLANet*, que consiste em duas sub-redes para o reconhecimento de alimentos. Uma sub-rede utiliza primeiro a atenção de canais espaciais híbridos para extrair características mais discriminatórias, e depois agrega essas características em uma representação de nível global (por exemplo, textura e informações de forma sobre alimentos). A outra gera regiões de atenção (por exemplo, regiões relevantes de ingredientes) de diferentes regiões através de transformadores espaciais em cascata, e agrega ainda mais estas características regionais de múltiplas camadas em representação a nível local. Estes dois tipos de características são finalmente fundidos para o reconhecimento de alimentos.

A tabela 3.2 mostra a acurácia alcançada para algumas base de dados neste estudo. Em comparação com métodos antigos, esse artigo atingiu o estado da arte.

Tabela 3.2: Resultados da arquitetura SGLANet para diferentes base de dados [23].

Dataset	**Acurácia Top-1**	**Acurácia Top-5**
FOOD 101	90.47%	98.21%
VIREO FOOD 172	90.78%	98.16%
ISIA FOOD 500	64.74%	89.12%

3.0.2 Segmentação semântica de alimentos

Conforme já mencionado nesse trabalho, segmentação semântica consiste em atribuir uma classe a cada pixel da imagem. Dessa forma, além da classificação de um alimento, é possível também identificar a posição do mesmo na foto. Essa é a forma mais precisa para calcular nutrientes e estimar valores nutricionais, já que a área do alimento na foto não tem o formato de um retângulo, mas sim seu formato real. Essa é a principal explicação pelo qual diversos estudos na área de *Food Computing* optam pela segmentação semântica ao invés da detecção de objetos com *Bounding Box*.

Da mesma forma que reconhecimento de alimentos é o passo inicial para qualquer aplicação nessa área, segmentá-los costuma ser o segundo passo. Apesar de nem todas as aplicações necessitarem de saber a posição e a área dos alimentos, a maioria que busca extrair dados úteis necessita dessa etapa. Será apresentado base de dados públicas e também os principais estudos dessa tarefa tão importante na construção de sistemas automáticos.

3.0.2.1 Datasets

Ainda utilizando a tabela 3.1, é possível extrair dois grandes *Datasets* com imagens anotadas para realizar segmentação semântica. Isso significa ter um fundo verdade com os valores do pixeis correspondentes à classe do alimento. [30] pegou uma parte de uma base de dados chamada FOOD201 e segmentou cada uma das fotos com 201 diferentes classes. No total, são 12.000 imagens. Outro estudo interessante é o UNIMIB16 [16], que coletou mais de 1.000 imagens em uma cantina na Itália e segmentou 73 diferentes tipos de alimentos. Alguns exemplos dessas imagens podem ser vistas na Figura 3.1.

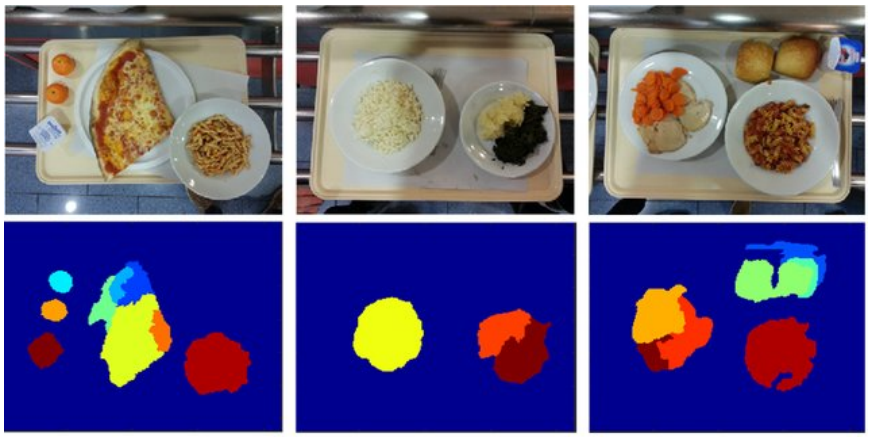


Figura 3.1: Base de dados UNIMIB16 para segmentação semântica [16].

Além das bases de imagens apresentadas no *Survey* [1], um estudo na China anotou imagens de forma semelhante aos *Datasets* já apresentados, como pode ser visto em [54]. Além dessas imagens usadas, o estudo também avaliou diferentes técnicas de segmentação semântica e usou essa base como *Benchmark* para comparação.

Outra base de dados recente é o UECFOODPIX [20], criado a partir do *Dataset* de reconhecimento de alimentos UEC256. Foram selecionadas aproximadamente 10.000 imagens e cada uma delas

foi segmentada detalhadamente. No total, são 102 tipos de alimentos predominantemente de cultura asiática. Atualmente, é o maior e mais completo compilado de imagens anotadas para realização de segmentação semântica. Semelhante a essa base e usando também imagens de alimentos da cultura oriental, foi lançando em 2021 o FoodSeg103 [3], com 103 classes de alimentos em mais de 9000 imagens. Em comparação com o UECFOODPIX, esse é mais desafiador e apresenta fotos mais complexas para se realizar a tarefa de segmentação semântica.

Por último, [21] foi o único conjunto de dados de alimentos brasileiros encontrados nas pesquisas. Isso é extremamente importante para o trabalho, já que serve de comparativo com o nosso *Dataset* criado. Ele conta com 9 classes de alimentos e 1250 imagens para treinamento, validação e teste.

3.0.2.2 Estudos recentes

A grande parte dos estudos relacionados a esse tema vêm dos próprios pesquisadores que criaram os *Datasets* mencionados na seção anterior. Por exemplo, [16] realizou seus testes no próprio UNIMIB16. Para a segmentação semântica, foi utilizado descritores visuais extraídos manualmente e um classificador SVM para atribuição das classes de cada pixel. Foi alcançado uma acurácia de 78.9% considerando as 73 possíveis classes de alimentos.

Já UECFOODPIX [20] usa uma abordagem mais recente: para extrair os resultados desejados, é utilizado uma rede neural convolucional DeepLab V3+ que possui uma arquitetura do tipo *Encoder Decoder*. Muito utilizada em segmentação semântica, os resultados alcançados foram uma acurácia de 66.8% e IoU de 55.5%. Outro trabalho que também utilizou a rede DeepLab é o estudo feito pelos autores de FOOD201 [30], onde foi encontrado uma acurácia de de 76% e IoU de 25%. Provavelmente o resultado desse trabalho foi pior devido ao *Dataset* utilizado ter mais classes disponíveis para o modelo escolher.

Como o artigo do FoodSeg103 [3] é bastante recente, foram utilizadas arquiteturas de *Deep Learning* bem recentes, como o uso de *Transformers* para codificar as informações da imagem. Apesar de serem modelos maiores e mais caros computacionalmente, os resultados encontrados superam o uso de CNNs tradicionais, como Resnet. Devido ao alto grau de complexidade das imagens do FoodSeg103, as métricas de avaliação da segmentação semântica foram baixas, com mIoU de 43.9% e acurácia de 57%. Em comparação com o estudo feito com a base de imagens de alimentos brasileiros [21] de apenas 9 classes e fotos controladas, este obteve mIoU de 70%, consideravelmente maior que o obtido pelo FoodSeg103. Isso mostra a dificuldade de obter sistemas robustos para segmentação semântica de alimentos, indicando a necessidade de muitos dados de treinamento para obter modelos confiáveis e capazes de classificar centenas de alimentos diferentes.

Mais recentemente, [55] desenvolveu uma nova abordagem para resolver o problema de segmentação semântica em alimentos e testa seus resultados tanto no UNIMIB16 quanto numa base dados privada. Eles propuseram uma arquitetura de CNN codificador-decodificador multiescala que compreende em uma microarquitetura codificadora residual, uma microarquitetura descodificadora em pirâmide e uma classificação especializada por pixel de alimentos/não-alimentos. Para os dados que utilizaram, conseguiram uma acurácia de 98.1% e IoU de 91.2%.

3.0.3 Estimativa de calorias e valores nutricionais

O problema de realizar estimativas de calorias e nutrientes presentes em uma refeição a partir de imagens está sendo abordado por diversos cientistas ao redor do mundo. Grande parte desse métodos tem como passo inicial a segmentação e classificação dos alimentos, como descrito anteriormente. Após realizar a segmentação semântica, a correlação com o valor nutricional de cada porção é obtida de diferentes maneiras. Duas das principais formas de alcançar tal objetivo são reconstrução 3D para estimação de volume e regressão polinomial para estimar nutrientes a partir de uma única foto.

3.0.3.1 Regressão polinomial

Essa abordagem consiste em aplicar funções polinomiais para correlacionar tamanhos de objetos presentes nas imagens e algum valor de interesse, como por exemplo, o valor calórico de tal porção. Os coeficientes do polinômio são obtidos muitas vezes a partir de treinamentos com imagens em que a quantidade de calorias está anotada junto a área de interesse. Esse treinamento geralmente é realizado para cada categoria de alimento separadamente, já que a variação desse valor de uma categoria para outra é abrupta. Também há casos em que a regressão é aplicada para correlacionar área e volume, com o objetivo de estimar o volume a partir de uma área na imagem.

Em abordagens como em [56], a correlação área - caloria é dada por uma regressão quadrática. A parte de segmentação também é abordada nesse artigo. Percebe-se que o método de segmentação *K-Means* (não supervisionado) é insatisfatório. Portanto, os autores optam por realizar a segmentação a partir do método *GrabCut*, utilizando de *bounding boxes* como supervisão para o treinamento, alcançando assim um resultado mais satisfatório.

$$C = ai * Fr^2 + bi * Fr + ci \tag{3.1}$$

Na equação 3.1 ai, bi e ci são constantes calculadas por categoria, no caso calculadas a partir de métodos de ajuste de curva polinomial. Fr corresponde a área da região ocupada pelo alimento em questão. Para realizar a estimativa de caloria a partir de uma única foto, os autores utilizam um objeto de tamanho conhecido (cartão pré-pago) para correlacionar as áreas na imagem com as áreas reais. Após isso, realiza-se uma regressão quadrática que correlaciona área e caloria, em que os coeficientes são obtidos para cada categoria de alimento separadamente.

Em [17], utiliza-se de duas fotos para realizar uma estimativa do volume da porção. A primeira foto deve ser uma vista superior do prato e a segunda uma vista lateral. Tendo ambas as perspectivas, calcula-se uma estimativa do volume a partir da altura e da área obtida pela rede de segmentação. Para correlacionar a área na imagem com a área real das porções, pede-se que, na foto superior, insira-se o polegar na imagem, de tamanho conhecido pelo algoritmo.

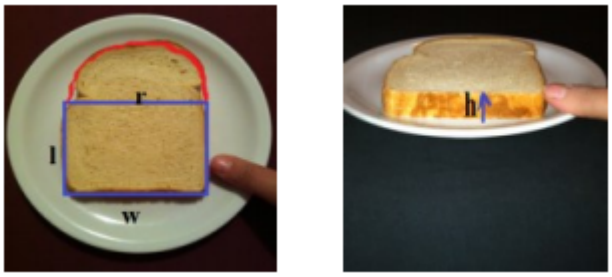


Figura 3.2: Método utilizado para estimativa de volume [17].

Os mesmos autores, em [57], inserem na proposta um servidor responsável pelo processamento das imagens, onde as redes neurais profundas estão hospedadas, retirando assim, a carga de processamento do telefone móvel e, por consequência, obtém-se um retorno mais rápido do algoritmo para o usuário final. Nessa solução, pede-se também como entrada do usuário o desenho do contorno da área a ser processada como imagem de alimento, reduzindo assim o ruído percebido pela rede ao classificar e segmentar as áreas.

3.0.3.2 Reconstrução 3D para estimativa de volume

Um método bastante utilizado para resolver o problema de estimativa de valores nutricionais e quantidade de calorias em alimentos é, de alguma maneira, obter o volume da porção que se deseja analisar e comparar com uma tabela de densidade de alimentos. Algumas delas são: [58] (Estados Unidos) e [59] (São Paulo - Brasil). A partir da densidade e do volume, pode-se obter os valores nutricionais desejados.

Para obtenção do volume das porções nas imagens, alguns trabalhos abordam o problema de maneiras distintas. Em [60], é proposto um sistema que recebe uma imagem RGB-D como entrada. A partir dessa imagem com valores também de profundidade, segmenta-se as porções 3D em subregiões de formato retangular alongado a fim de realizar um procedimento similar a uma integral, ou seja, calcular o volume para cada uma dessas subregiões a partir da sua área e profundidade e somar todas elas para o cálculo do volume da porção como um todo.

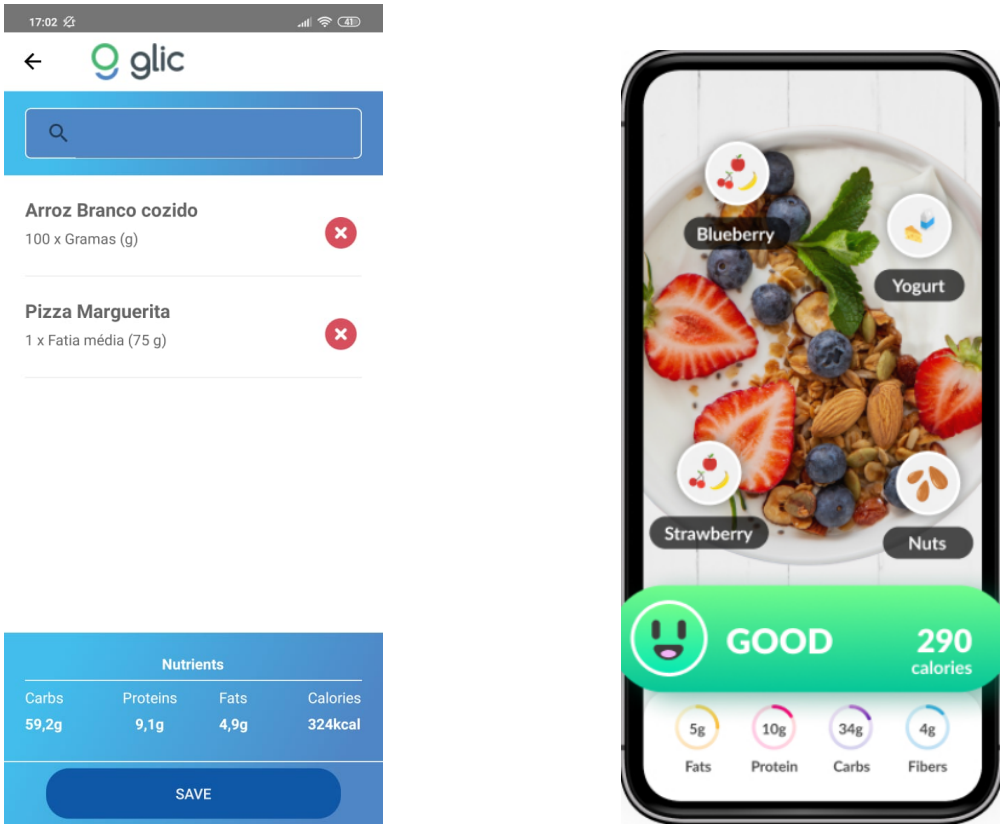
Esse método foi testado em dispositivos mobile com kits de realidade aumentada embutidos. Telefones celulares com câmeras traseiras duplas, como o Iphone X ou Samsung S10 dispõem dessa funcionalidade e foram utilizados no artigo em questão.

Como o requisito tecnológico pode ser uma barreira muito grande para aderência de muitos usuários, uma outra abordagem presente na literatura se baseia em sistemas de visão estéreo, com requisito de hardware menos estrito. Em [61], telefones com apenas uma câmera traseira devem retirar um par de fotos que, com o auxílio da aplicação, se encontram a 90º e 75º do plano do prato em análise. Tendo essas duas vistas disponíveis, o algoritmo realiza a calibração extrínseca do par de câmeras. Após isso, a reconstrução tridimensional densa é realizada seguindo os passos: retificação das imagens, correspondência estéreo e criação da nuvem de pontos. Uma segunda opção suportada é gravar um curto vídeo, onde os *frames* com melhor angulação serão capturados e processado pelo algoritmo descrito anteriormente.

3.0.4 Aplicativos *mobile* no mercado

A demanda por ferramentas que auxiliem na dieta alimentar é grande, tendo em visto a crescente necessidade de controlar sobrepeso e auxiliar o tratamento de doenças, como Diabetes. Isso foi observado no grande número de estudos relacionados a esse tema, assim como um extenso *Survey* [1] para resumir toda essa informação. E não seria diferente fora do mundo acadêmico: hoje é possível encontrar nas lojas de aplicativos de celulares diversas aplicações que buscam resolver vários dos problemas abordados. O principal deles parece ser a estimação de nutrientes, já que essa informação é a que agrega mais valor aos pacientes e médicos que buscam analisar a ingestão de alimentos no dia a dia.

Existem alguns aplicativos que realizam essa tarefa de forma simples e bastante manual, sem nenhuma técnica avançada de engenharia para automatizar processos. Por exemplo, Glic [18] é uma ferramenta que auxilia nas tarefas diárias de quem tem diabetes, entre elas a contagem de carboidratos e outros nutrientes que impactam a glicemia do paciente. Como grande parte dos diabéticos insulino-dependentes necessitam calcular uma dose de insulina baseada na quantidade de carboidratos ingeridos, tal aplicativo se mostra extremamente útil para dar tal informação. O usuário tem um campo de busca onde ele digita o nome do alimento e a porção que consumiu (veja Figura 3.3). No final, ele tem um relatório completo de tudo que consumiu e pode, assim, tomar decisões em cima disso.



(a) Glic (b) *Foodvisor*

Figura 3.3: Aplicativos *mobile* [18] e [19].

Além do controle de diabetes, calcular quantas calorias uma pessoa ingere por dia parece ser uma tarefa ainda mais buscada por pessoas de diversas faixas etárias. Um exemplo de aplicativo móvel que realiza tal tarefa com êxito é o *MyFitnessPal* [62]. Ele funciona de forma parecida com o *Glic*, porém com foco e métricas voltadas para emagrecimento e qualidade da alimentação.

Mas com os avanços das técnicas já mencionadas nos trabalhos relacionadas, é evidente que aplicações mais robustas e automáticas surjam no mercado. Um exemplo disso é o *CalorieMama* [63], também focado na contagem de calorias e no emagrecimento e boa forma dos seus usuários. Porém, uma grande diferença é que, ao invés de buscar numa base de dados os alimentos ingeridos de forma manual, basta tirar uma foto do prato de alimento e os alimentos serão reconhecidos automaticamente, sendo necessário apenas inserir a porção consumida.

Outra opção bastante parecida é o *Foodvisor* [19], que realiza basicamente as mesmas tarefas da aplicação mencionada anteriormente e estima os nutrientes baseado numa porção média do alimento, e não baseado no conteúdo da imagem (veja Figura 3.3). Uma coisa fica clara: não se tem conhecimento de nenhuma ferramenta no mercado que realmente estime os nutrientes ou o peso dos alimentos por meio de uma foto. Nesse cenário, não seria preciso que o usuário tivesse quase nenhuma ação além de tirar fotos dos alimentos. Isso mostra uma oportunidade para o desenvolvimento deste trabalho e também caracteriza o quão desafiador pode ser implementá-lo com robustez.

Capítulo 4

Metodologia

Neste capítulo será detalhado com minuciosidade todos os passos realizados nesse trabalho, desde a concepção da base de dados até a implementação do algoritmo e do ambiente de testes. O código utilizado foi anexado e pode ser consultado no final do documento, no Anexo I. O link de acesso do Github do projeto pode ser encontrado na referência [64] e o banco de dados criado também pode ser acessado a partir da referência [24].

4.1 Elaboração da Base de Dados de Imagens

4.1.1 Análise dos hábitos alimentares

A seleção dos alimentos que fazem parte da base de dados baseou-se na pesquisa realizada em [65], onde 94% dos entrevistados afirmaram comer, no almoço, arroz e feijão acompanhados de carne vermelha (69%), galinha (42%), salada (30%), macarrão (24%), verduras em geral (22%) e legumes (18%). Esses valores variam de acordo com a renda e com a localidade, porém, na média esses são os alimentos mais consumidos pelos brasileiros em sua refeição principal. Com o objetivo de abranger mais localidades e culturas diferentes, incluiu-se algumas categorias de importância regional como peixes e frutas.

4.1.2 Levantamento das Principais Bases de Dados Públicas de Alimentos

A fim de comparar os resultados alcançados por outros trabalhos, buscou-se bases de dados de segmentação de alimentos. Houve uma enorme dificuldade de encontrar trabalhos relacionados a alimentação brasileira, com exceção de [21]. Outra base importante para o trabalho é o UECFO-ODPIX [20] que conta com 10.000 imagens (9.000 treinamento e 1.000 teste) de comidas de origem japonesa segmentadas por alimento. Também foram utilizadas bases de outras nacionalidades como o UNIMIB16 [16] de comida italiana e o FoodSeg103 [3] de Singapura.

4.1.2.1 UECFoodPix

O banco com 10.000 imagens realizou a divisão treinamento/teste em 90%/10%. A base de imagens japonesa possui 103 classes, sendo que uma delas está destinada ao plano de fundo. Exemplo de uma imagem anotada do banco 4.1.

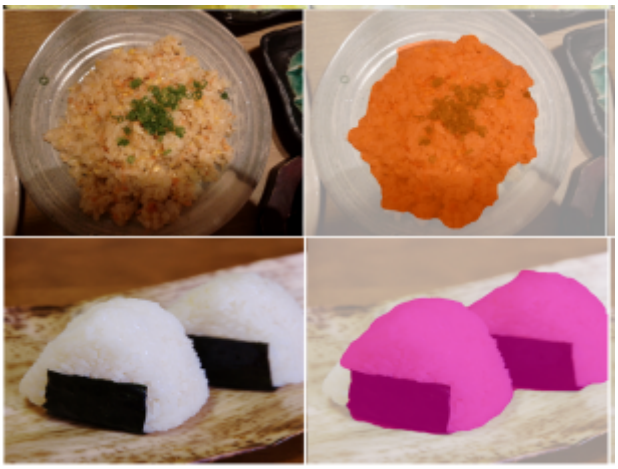


Figura 4.1: Exemplo de segmentação UECFOODPIX [20].

4.1.2.2 FoodSeg103

Este banco de imagens conta com 7.118 imagens e utilizou um percentual de divisão treinamento/teste 70%/30% aproximadamente com 4.983 imagens para treinamento e 2.135 imagens para teste. Essa base conta com 104 classes, sendo uma delas destinada ao plano de fundo e uma a uma classe de alimentos genérica (*"other ingredients"*). Exemplo de imagem anotada em 4.2.

Figura 4.2: Exemplo de segmentação FoodSeg103 [3].

4.1.2.3 UNIMIB16

Este banco de imagens conta com 1.010 imagens e utilizou um percentual de divisão treinamento/teste 64%/36% aproximadamente com 650 imagens para treinamento e 360 imagens para teste. Possui 74 classes, uma destinada ao plano de fundo. Exemplo de imagem anotada em 4.3.

Figura 4.3: Exemplo de segmentação UNIMIB16 [16].

4.1.2.4 MyFood DataSet (PEFoods)

Este banco de imagens conta com 1.250 imagens e utilizou um percentual de divisão treinamento/teste 60%/40% aproximadamente com 750 imagens para treinamento e 500 imagens para teste. O banco de imagens de brasileiro conta com 11 classes, uma para "NãoAlimentos"e uma para "NãoRegistrado". Exemplo de imagem anotada em 4.4.

(a) Imagem (b) *Ground-Truth*

Figura 4.4: Exemplo de Imagem Anotada MyFood. Nota-se um nível de cinza bem escuro representando o alimento, sendo quase imperceptível a olho nu [21].

4.1.3 Coleta de Imagens de Alimentos para Construção da Nova Base BR-UMAS21

A construção da base de comidas brasileiras segmentadas necessitou uma busca de forma automática dada a quantidade de imagens necessárias ser grande. Essa busca foi realizada através de uma ferramenta (*crawler*) que realiza requisições a sites de busca de imagens como *GoogleImageSearch* e *BingImageSearch* a partir de palavras-chave. A ferramenta realiza as requisições e salva as imagens em arquivo local. Porém, a utilização de uma ferramenta automática tem como aspecto negativo a ocorrência de fotos duplicadas. Para solucionar esse problema, um algoritmo de DHASH 2.1.1 foi utilizado para a comparação e exclusão de fotos duplicadas.

A partir desse processamento foi feita uma limpeza, retirando imagens com angulação ruim e com alimentos que não estavam previstos para compor a base de imagens. Antes, a base contava com 2.000 imagens e após o procedimento para deixá-la mais concisa e enxuta conta com 512 imagens (figura 4.5).

Figura 4.5: Exemplos de imagens na base BR-UMAS21 I.1.

4.1.4 Definições de Classes e Agrupamentos Alimentares

Analisando as imagens coletadas, retirou-se classes que possuíam menos de duas imagens, enxugando a base para 469 imagens com 77 classes. A partir deste cenário inicial (Baseline Sprint 0), foi aplicada uma estratégia de variação da granularidade das classes, mudando os agrupamentos para comparar resultados.

4.1.4.1 Baseline - Sprint 0

O treinamento inicial foi realizado com classes organizadas de forma granular e detalhista, ou seja, variações de um mesmo alimento possuem classes próprias. A tabela 4.1 e o histograma 4.6 mostram a distribuição das classes.

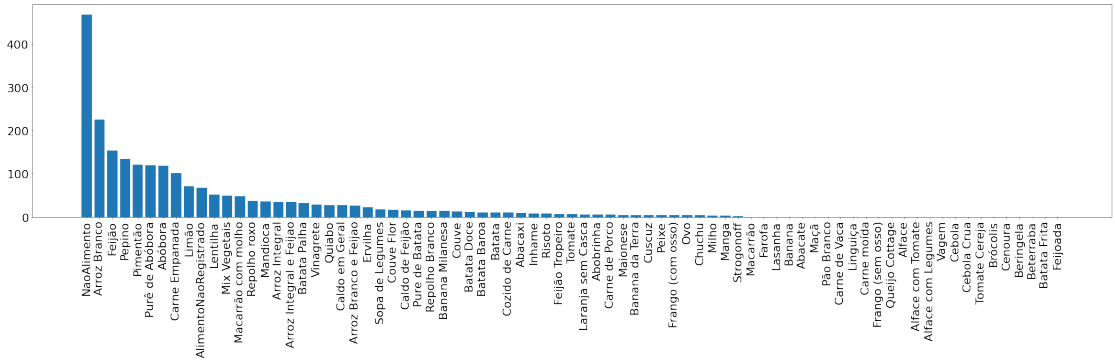


Figura 4.6: Histograma de imagens por classe - Sprint 0.

Tabela 4.1: Classes referentes ao Sprint 0.

ClassID	ClassName	ClassID	ClassName
0	NaoAlimento	62	Queijo Cottage
1	AlimentoNaoRegistrado	65	Alface
2	Arroz Integral	66	Alface com Tomate
3	Arroz Branco	67	Alface com Legumes
4	Feijão	68	Vagem
5	Arroz Integral e Feijao	69	Cebola
6	Arroz Branco e Feijao	70	Cebola Crua
8	Macarrão	71	Tomate Cereja
9	Batata	72	Tomate
10	Batata Palha	73	Brócolis
11	Batata Doce	74	Cenoura
12	Pure de Batata	75	Abobrinha
13	Mandioca	76	Beringela
14	Inhame	77	Beterraba
16	Cuscuz	78	Abóbora
17	Farofa	80	Couve
18	Feijão Tropeiro	81	Repolho Branco
20	Lentilha	82	Couve Flor
21	Batata Baroa	83	Quiabo
22	Lasanha	84	Ervilha
24	Risoto	85	Sopa de Legumes
27	Milho	88	Pimentão
28	Maionese	89	Mix Vegetais
31	Banana	90	Chuchu
32	Manga	91	Batata Frita
35	Abacate	92	Feijoada
36	Maçã	93	Pepino
38	Abacaxi	95	Repolho roxo
47	Laranja sem Casca	96	Macarrão com molho
50	Pão Branco	98	Banana Milanesa
53	Carne de Vaca	99	Carne de Porco
54	Cozido de Carne	100	Banana da Terra
55	Strogonoff	102	Caldo de Feijão
56	Linguiça	104	Purê de Abóbora
57	Carne moída	106	Limão
58	Peixe	109	Carne Empanada
59	Frango (com osso)	111	Vinagrete
60	Frango (sem osso)	112	Caldo em Geral
61	Ovo		

4.1.4.2 Sprint 1

A fim de diminuir o número de classes e aumentar o número de amostras por classe, utilizou-se uma outra forma de agrupamento (Sprint 1), agora em 17 classes mais genéricas que abrangem mais de um tipo de alimento. A tabela 4.2 e o histograma 4.7 mostram a distribuição das classes.

Tabela 4.2: Classes referentes ao Sprint 1.

ClassID	ClassName
0	NaoAlimento
1	AlimentoNaoRegistrado
118	TiposArroz
119	Leguminosas
120	Frutas
121	Legumes
122	Batatas
123	Pure
124	Salada
125	CarneAnimal
126	Massa
127	Queijos
128	Farofa
129	Ovo
130	Paes
131	Torta
132	Caldos

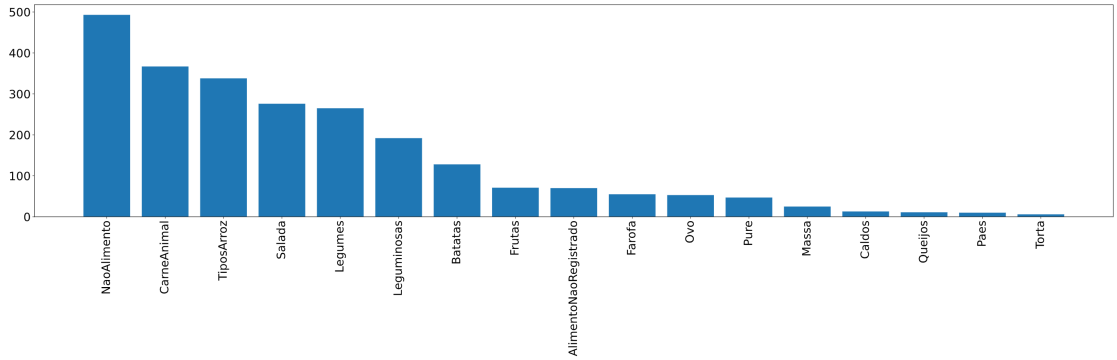


Figura 4.7: Histograma de imagens por classe - Sprint 1.

4.1.4.3 Sprint 2

Por fim, focou-se em um agrupamento (Sprint 2) em grandes grupos de alimentos de acordo com valor nutricional, tornando a base ainda menos granular e com apenas 5 classes diferentes. A tabela 4.3 e o histograma 4.8 mostram a distribuição das classes.

Tabela 4.3: Classes referentes ao Sprint 2.

ClassID	ClassName
0	NaoAlimento
117	Carbs
118	Proteina
119	Vegetais
120	Frutas

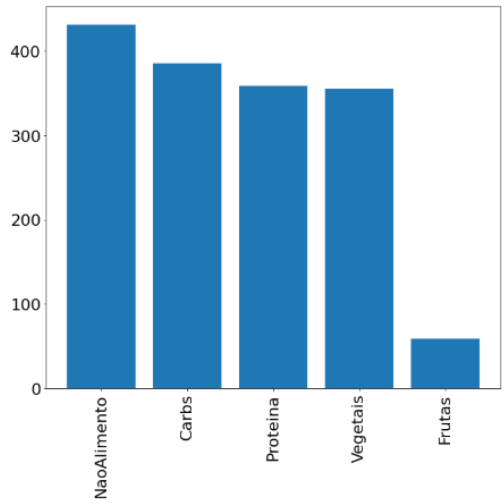


Figura 4.8: Histograma de imagens por classe - Sprint 2.

Em todos os cenários, o número de amostras por classe estava desbalanceado e foram utilizadas técnicas de *Data Augmentation* vistas na subseção 2.1.3 para tentar minimizar o dano causado ao treinamento. Além disso, o número total de imagens varia para cada um dos agrupamentos devido a amostras cujas categorias não puderam ser adicionadas a um grupo maior. Um exemplo são as categorias "bolo de chocolate"e "ketchup"que foram retiradas no segundo agrupamento.

Outro ponto importante a se explicitar é a estratégia de retirar a classe "AlimentoNãoRegistrado"para o último sprint (sprint 2). Essa estratégia foi utilizada para excluir uma possível fonte de ruído que estaria acontecendo devido a aglomeração de muitas imagens com características diferentes em uma mesma classe.

Todas as imagens foram rotuladas utilizando o agrupamento mais granular, com 113 classes diferentes. Isso foi realizado, pois mudar os rótulos para categorias menos específicas pode ser feito em código e não é necessário rotular novamente as imagens, apenas redefinir os valores para os rótulos. O contrário não seria possível.

4.1.5 Apresentação da Estratégia de Rotulamento da Base Construída

Os rótulos foram feitos utilizando uma ferramenta própria, construída em C++ e utilizando a biblioteca de visão computacional OpenCV. A ferramenta utiliza como entrada traços realizados pelo usuário para definir o contorno das áreas desejadas. A partir desses traços de entrada, utiliza-se a técnica de *WaterShed* 2.1.2 para preencher a área de forma a respeitar ao máximo o limite imposto pelo usuário e ser compatível com a área apresentada na imagem (figura 4.9).

(a) Imagem com Traços de Entrada.

(b) *Imagem após WaterShed.*

Figura 4.9: Entrada do usuário e resultado WaterShed.

Tendo em mãos a imagem de retorno do algoritmo de *WaterShed*, associa-se a cada área uma classe. Para relacionar uma área a uma classe, cria-se uma máscara em escala de cinza (*ground-truth*) onde cada pixel tem um valor de uma respectiva classe. Abaixo (figura 4.10) é possível ver a correlação área-classe.

(a) Imagem de alimentos.

(b) *GroundTruth*.

Figura 4.10: Imagem e GroundTruth.

Em comparação com outras ferramentas de anotação, a ferramenta própria apresenta algumas vantagens. Geralmente, as ferramentas disponíveis para essa tarefa utilizam marcação poligonal para definir as áreas, ou seja, cada clique define um vértice do polígono (figura 4.11). Esse método necessita de mais tempo para anotar cada imagem se compara a ferramenta utilizada, já que necessita de vários cliques com precisão. Além disso, por ser uma área poligonal, muitas vezes não coincide muito bem com a área da imagem, dificultando o processo e requerendo pontos muito próximos uns dos outros.

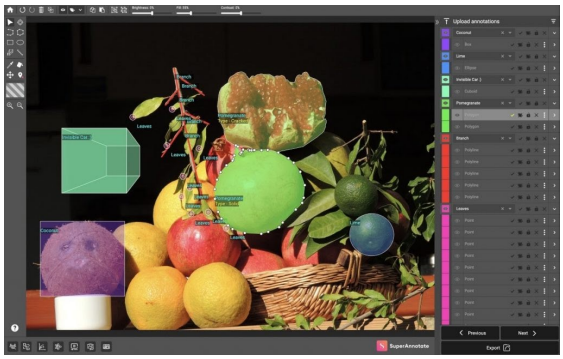


Figura 4.11: Ferramenta de anotação [22].

Percebe-se que para áreas simples, como um círculo, a ferramenta que utiliza pontos de polígono apresenta uma facilidade grande. Entretanto, se utilizada para áreas mais complexas, a delimitação através de pontos fica mais difícil se comparada a ferramenta própria. Por outro lado, um aspecto negativo da ferramenta utilizada nesse trabalho é que o algoritmo de WShed pode apresentar erros ao definir os contornos, porém pode ser corrigida com mais um traço de entrada do usuário anotador.

4.2 Descrição do algoritmo de segmentação semântica

O algoritmo, que se encontra no final deste documento, utilizou o Google Colab como ambiente de execução. Utilizou-se da ferramenta para diminuir o tempo de treinamento, já que conta com um poder computacional muito maior que computadores pessoais. A GPU disponibilizada pela plataforma na sua versão gratuita é Nvidia Tesla T4, com 15109MiB de memória. Em alguns cenários de teste, onde a carga de processamento foi acima da média (Ex: algoritmo de *cross-validation*), foi necessário adquirir a versão paga da plataforma pra ter acesso a uma placa de vídeo mais potente. No caso, a GPU fornecida pela plataforma paga foi Tesla P100-PCIE, com 17071 MiB de memória.

Como ferramenta para atacar o problema descrito, a biblioteca fastai v2 [7] foi utilizada. Tal biblioteca está se mostrando referência no mercado para facilitar o acesso as práticas de *Deep Learning*. Por se tratar de uma biblioteca que tem como base o PyTorch, é bastante robusta e dada sua arquitetura em camadas permite uma programação em mais alto nível. Dado tal contexto, algumas configurações padrão utilizadas no âmbito de treinamento da rede são:

- Função de Otimização - Adam (2.3.2)
- Função de Custo - Focal Loss (2.3.1)
- *Backbone* da CNN - Resnet34 (2.2.2)
- Arquitetura da rede - Unet (2.2.3)

4.2.1 Carregamento dos Dados e Data Augmentation

Para realizar o processo de treinamento da rede, a base conta com duas pastas principais, uma com as imagens e outra com os rótulos. Além disso, há um arquivo texto que correlaciona o nome da classe e seu número, como visto na subseção 4.1.4.

A estratégia de variar a granularidade das classes necessitou de um remapeamento dos rótulos. Por exemplo, classes como Arroz Branco, Arroz Integral, arroz branco e feijão, arroz integral e feijão foram remapeadas para a classe TiposArroz, no Sprint 1. Devido a essa necessidade, aplica-se um algoritmo de remapeamento logo após o carregamento dos dados.

Após o remapeamento, define-se as técnicas de aumento de dados vistos na subseção 2.1.3, na biblioteca [7] fastai v2.0. Escolheu-se técnicas que distorcem pouco as imagens a fim de evitar perdas de características importantes para a segmentação. Por tanto, foram utilizadas apenas técnicas de rotação, zoom, transformações de luz e *warp*. Outra estratégia adotada para tentar minimizar o desbalanceamento da base de dados foi aplicar pesos para as diferentes classes, isto é, classes que aparecem menos têm um peso maior que classes que aparecem mais. Apesar de ser teoricamente coerente, os resultados demonstraram efeito quase nulo para o peso inversamente proporcional ao número de aparições e por isso a técnica não foi mais utilizada.

APIs que tratam os dados foram necessárias. As interfaces DataBlock e DataLoader da biblioteca Fastai v2 [7] permitem aplicar técnicas de aumento de dados durante o treinamento além de

separar as imagens de treinamento e as imagens de teste. A separação treinamento-teste foi realizada com 80% para treinamento e 20% para teste. Essa separação, porém, pode fazer com que imagens de uma mesma classe se acumulem no conjunto de teste e a rede não seja estimulada o suficiente para determinada classe. Para contornar este problema, se propõe um algoritmo de estratificação por classe, que tenta manter além dos 80-20 globais, a mesma separação interna das classes.

4.2.2 Modelo e arquitetura da rede

Utilizou-se como base uma Resnet34 pré-treinada com as imagens do banco de imagens IMAGENET. Como algoritmo de segmentação semântica foi utilizado Unet. Em relação a parte de função de custo, utilizou-se *Focal Loss* visto na subseção 2.3.1, já que essa função varia o peso das classes dinamicamente de acordo com sua dificuldade, sendo adequada para o treinamento utilizando classes desbalanceadas.

4.2.3 Treinamento

O primeiro passo é definir o valor da taxa de aprendizagem (*learning-rate*). Para isso, o fastai possui uma função que avalia a *loss* em função da taxa de aprendizagem. Essa avaliação busca pela maior variação da *loss* em função do *learning-rate* (derivada da função) para sugerir o que seria a melhor taxa de aprendizagem (figura 4.12). O segundo passo é definir a quantidade de épocas. No caso, realiza-se 10 a 15 épocas com os pesos dos neurônios das camadas convolucionais congelados, para atualização da camada densa final, e 25 épocas permitindo que toda a rede tenha seus pesos atualizados. Após todos esses passos é possível treinar a rede.

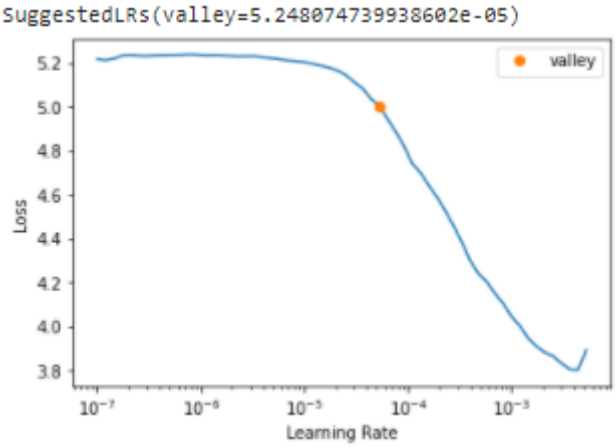


Figura 4.12: Learning Rate vs Loss I.1.

4.3 Métricas de Avaliação do Processo de Segmentação

As métricas para avaliar o processo de segmentação diferem um pouco das métricas utilizadas para avaliar um processo de classificação. Uma das métricas que exemplifica isso é a acurácia por pixel. Essa métrica leva em conta quantos pixels foram preditos de forma correta em relação

ao rótulo, porém não leva em conta as classes e nem suas respectivas áreas. Por exemplo, em imagens que tem 90% da sua área ocupada por uma classe denominada "plano de fundo", não basta que a rede acerte 90% dos pixels, a rede deve ser avaliada quanto ao acerto por classe e é por isso que se utiliza métricas como mIoU mostrada na subseção 2.4.2 e F1-Score mostrada na subseção 2.4.3, além da acurácia para segmentação vista na subseção 2.4.4, em que o acerto do "plano de fundo"é desconsiderado. Além de utilizar tais métricas, utilizou-se o método de validação cruzada (*cross-validation*) com 5 diferentes agrupamentos para reduzir o viés da rede.

4.3.1 Elaboração dos Cenários de Teste e Avaliação dos Modelos e Bases de Imagens

Tendo em vista as métricas utilizadas para comparar a efetividade da segmentação da rede em relação com os resultados obtidos por outros estudos e bases públicas, foram realizados os seguintes cenários de teste. Para todos os cenários de teste utilizou-se os seguintes hiper-parâmetros.

- Tamanho das imagens (256, 256)
- *Weight Decay* - 0.01
- *Learning-rate* - 1e-4
- *Batch Size* - 8

Primeiro, validou-se o fluxograma apresentado na seção 4.2 e a arquitetura/modelo da rede apresentado na subseção 4.2.2 aplicando o treinamento para os bancos de imagens UECFOODPIX [20], UNIMIB16 [16], FOODSEG103 [3] e MyFood/PE_Foods [21]. Nesses treinamentos, foi seguido à risca a divisão que os autores dos artigos fizeram dos dados de treinamento, validação e teste.

Após validado, treinou-se o modelo utilizando três diferentes tipos de agrupamento (Sprint 0, Sprint 1 e Sprint 2) da base própria BR-UMAS21. Essa comparação manteve todos os hiperparâmetros fixos e variou apenas a granularidade das classes para analisar seu efeito. Os hiperparâmetros fixos utilizados para os testes foram descritos acima. Apenas em um dos casos, no Sprint 2, variou-se também a arquitetura ou *backbone* da rede. Foram testadas as redes Resnet34, Resnet101, VGG16 e AlexNet.

Ao aplicar o fluxograma para os dados da base BR-UMAS21, utilizou-se também da técnica de *cross-validation* explicada na subseção 2.4.1. A base de imagens foi separada em 60/20/20, o que significa que 20% são dados fixos de teste e 20% variará a cada iteração do algoritmo para validação. Ou seja, cada vez separa-se aleatoriamente quais amostras estarão no conjunto de 60% e quais estarão no subconjunto de 20% para compor o conjunto de validação.

Capítulo 5

Resultados

Utilizando o algoritmo de segmentação proposto no capítulo de Metodologia 4 e a biblioteca Fastai [7] para implementação do modelo *Deep Learning*, foi possível variar apenas o *dataset* de entrada para encontrar as métricas de avaliação (também descritas na seção 4.3). O código utilizado para obter os resultados foi anexado e pode ser consultado no final do documento, no Anexo I. O link de acesso do Github do projeto pode ser encontrado na referência [64] e o banco de dados criado também pode ser acessado a partir da referência [24].

5.1 Avaliação do Modelo para as Bases de Dados Públicas

5.1.1 UECFoodPix

Mantendo a mesma proporção dos dados de treinamento e teste, proposto no artigo [20] (detalhes na subseção 4.1.2.1), foi possível comparar o resultado obtido pelo nosso modelo e pelo próprio artigo, conforme pode ser visto na tabela 5.1. Observa-se que o modelo implementado superou os resultados obtidos no estudo [20], muito provavelmente porque a arquitetura Unet se adequou melhor aos dados e comparação com DeepLab v3+. As técnicas de *Data Augmentation* utilizadas no nosso modelo também foram bem eficientes.

Tabela 5.1: Comparação dos resultados para a base de dados UECFOODPIX.

Arquitetura	**Acurácia**	**mIOU**
Unet com Resnet34 (nosso)	**71%**	**59%**
DeepLab v3+ (artigo UECFOODPIX)	66,8%	55,5%

5.1.2 FoodSeg103

O mesmo foi feito para o *dataset* FoodSeg103 [3] (detalhes na subseção 4.1.2.2). A tabela 5.2 mostra o comparativo dos resultados. Nesse caso, nosso algoritmo não foi capaz de superar

o resultado obtido em [3], pois o modelo que apresentou melhor resultado possui mecanismos de atenção *Transformers*, que vem ganhando muita força nos últimos anos. Modelos de visão baseados em *Transformers*, como ViT, estão atingindo estado da arte em várias tarefas, incluindo a segmentação semântica de alimentos. Além disso, outra hipótese para a baixa performance é que o nosso modelo poderia ter sido treinado mais épocas ou ter sido usado uma arquitetura Resnet mais profunda, como Resnet101.

Contudo, se compararmos o resultado de mIOU (principal métrica de avaliação) obtido pelo modelo FPN com Resnet50, nota-se que o resultado foi bem parecido, mostrando que nosso modelo não destoou muito de outras arquiteturas similares.

Tabela 5.2: Comparação dos resultados para a base de dados FoodSeg103.

Arquitetura	**Acurácia**	**mIOU**
Unet com Resnet34 (nosso)	52%	26%
FPN com Resnet50 (artigo FoodSeg103)	38,2%	27,8%
SeTR com ViT-16/B (artigo FoodSeg103)	**52,7%**	**41,3%**

5.1.3 UNIMIB16

Apesar de [16] ser um estudo mais antigo, de 2016, o banco de imagens construído continua sendo relevante para avaliar modelos de segmentação semântica. Este foi criado em ambientes controlados e cuidadosamente anotado, portanto espera-se que os resultados sejam superiores em relação a UECFOODPIX e FoodSeg103. A tabela 5.3 mostra a comparação dos resultados obtidos, revelando mais uma vez que o nosso modelo superou o implementado no artigo. Os dados foram organizados conforme descrito na subseção 4.1.2.3.

Tabela 5.3: Comparação dos resultados para a base de dados UNIMIB16.

Arquitetura	**Acurácia**	**mIOU**
Unet com Resnet34 (nosso)	85%	64%
SVM + AlexNet (artigo UNIMIB16)	78,9%	-

Mesmo que a métrica mIOU encontrada nesse caso (mIOU = 64 %) supere o resultado de UECFOODPIX (mIOU = 59%), percebe-se que ainda não foi encontrado um modelo bom o suficiente para colocar um sistema de reconhecimento de alimentos em produção, já que mIOU abaixo de 75% não costuma ser confiável. Uma possível explicação para isso é o alto número de classes de alimentos nesses *datasets* (entre 70 e 100 classes), aliado ao baixo número de imagens dos bancos (entre 1k e 10k imagens). Dessa forma, não há nenhum modelo encontrado capaz de aprender esta tarefa tão complexa.

Outra possível resposta para tamanha dificuldade em reconhecer os alimentos é o desbalanceamento das classes do banco, já que classes que aparecem menos são naturalmente mais difíceis de aprender e podem reduzir a métrica. Além disso, há uma dificuldade inerente na tarefa de segmentação semântica de alimentos: frequentemente aparecem alimentos difíceis de serem identificados, até por um ser humano. Nesses casos (não tão raros), o sistema se confunde e reduz bastante a métrica de avaliação.

Mesmo nesse *dataset* limpo, bem organizado e razoavelmente balanceado, isso acontece. A figura 5.1 mostra que alimentos comuns e com características claras possuem predição excelente (macarrão, vagem e couve tiveram predição parecida com o fundo verdade). Já o alimento verde não teve uma saída desejada, e com razão: após analisar a imagem, não conseguimos determinar se esse alimento é frango ou batata, explicando porque a métrica mIOU foi bastante reduzida nesse exemplo. É interessante pontuar que esse tipo de imagem ocorre várias vezes e explica porque a saída da rede neural foi ruim. Talvez com mais dados de treinamento e um banco mais robusto, o modelo fosse capaz de escolher entre os dois alimentos (batata e frango), ao invés de predizer vários alimentos diferentes.

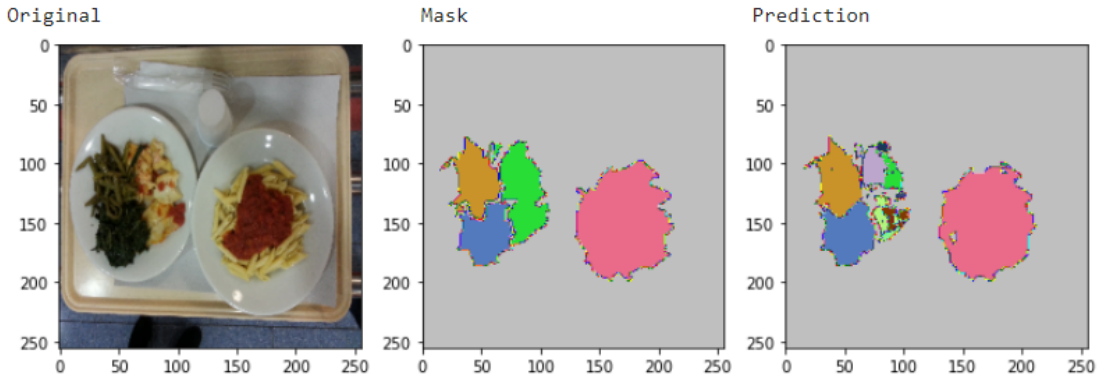


Figura 5.1: *Mask* representa o fundo verdade esperado, enquanto *Prediction* mostra a saída do nosso modelo. Nota-se que a detecção do alimento da máscara verde foi ruim, porém os outros tiveram resultados aceitáveis.

5.1.4 MyFood

Esta, talvez, é a base de dados mais interessante de se testar, já que foi criada por brasileiros, com o intuito de conter alimentos típicos da nossa culinária. Apesar do entusiasmo, a base contém apenas 9 classes de alimentos (mais detalhes na subseção 4.1.2.4), o que facilita bastante a tarefa de segmentação semântica e não representa bem os dados esperados por um sistema que opera no mundo real. Os resultados confirmam essa hipótese, conforme mostra a tabela 5.4:

Tabela 5.4: Comparação dos resultados para a base de dados MyFood.

Arquitetura	Acurácia	mIOU
Unet com Resnet34 (nosso)	88%	85%
MASK RCNN com Resnet101 (artigo MyFood)	-	70%

Nosso modelo foi mais uma vez capaz de superar os resultados do estudo MyFood [21]. Mesmo que o artigo seja recente e use arquiteturas atualizadas e capazes de serem estado da arte (como por exemplo MASK RCNN), nosso algoritmo leva vantagem por ser implementado com a biblioteca Fastai [7]. Esta é muita coesa e usa técnicas avançadas para treinar modelos de forma rápida, como o Fit-one-cycle. Porém, acredita-se que o grande diferencial sejam os métodos de *Data Augmentation* padrões da biblioteca, o que faz o modelo generalizar melhor, e a implementação personalizada da arquitetura UNET, que é bastante eficiente e otimiza os resultados.

5.2 Avaliação do Modelo para a Nova Base BR-UMAS21

Resumindo os resultados da seção anterior 5.1, nota-se que o algoritmo de segmentação implementado foi bastante eficiente, superando 3 de 4 artigos que criaram as bases de dados públicas. Tendo isso em visto, é seguro testar o mesmo algoritmo com o *dataset* de alimentos típicos brasileiros.

Recordando o capítulo de Metodologia 4, o *dataset* BR-UMAS21 foi organizado de três formas, cada uma sendo representada por um Sprint. Cada Sprint agrupa as classes dos alimentos em níveis granulares diferentes e os detalhes podem ser revistos na subseção 4.1.4. Além disso, foi utilizado *Cross-Validation* em todos os Sprints e manteve-se o algoritmo de segmentação usado na validação das bases públicas..

5.2.1 Sprint 0

O primeiro cenário de testes envolve pouca ou quase nenhuma alteração na base de dados originalmente anotada. Conforme visto na subseção 4.1.4.1, Sprint 0 manteve as classes originais de cada alimento.

Usando a técnica de *Cross-Validation* para evitar qualquer viés inerente aos dados, foi possível verificar o desempenho do algoritmo de segmentação usando UNET com Resnet34 nas 477 imagens, sendo 60% delas usadas no treinamento, 20% na validação e 20% nos testes. A tabela 5.5 a seguir mostram os resultados obtidos seguindo o algoritmo já descrito na seção 4.2.

Tabela 5.5: Validação cruzada para os dados do Sprint 0.

Cross-Validation Fold	Dataset	Acurácia	mIOU
1	Validação	47,7%	14,6%
	Teste	49,1%	16,8%
2	Validação	50,2%	14,6%
	Teste	46,8%	15,9%
3	Validação	48,2%	14,9%
	Teste	49%	15,3%
4	Validação	52,4%	15,1%
	Teste	48,6%	17,1%
5	Validação	46,1%	12,88%
	Teste	48%	16,3%
Média	Validação	48,9%	14,4%
	Teste	48,3%	16,3%

Avaliando apenas os dados de teste, mIOU médio foi de apenas 16,3%, um valor muito baixo. Isso mostra um caso típico de *underfitting*, onde o modelo não foi capaz de aprender a tarefa com os dados de treinamento usados. Fazendo uma breve comparação com os outros *datasets* públicos vistos na seção 5.1, nota-se que os mIOUs encontrados também foram baixos, porém superiores a 16,3%. Isso ocorreu muito possivelmente devido a qualidade e quantidade dos dados. Nossa base de dados tinha apenas 477 imagens para 77 classes de alimentos, uma proporção muito baixa. Já as bases públicas mencionadas estão na faixa de 7-10k imagens para 100 classes de alimentos.

Outro fator que contribui para abaixar a qualidade dos dados é a classe de alimentos "Alimento-NaoRegistrado". Essa classe engloba todo alimento que destoa do padrão e é difícil de identificar. Por exemplo, uma berinjela sobreposta por queijo e tomate não pode ser classificada como berinjela, pois seu aspecto visual se difere de uma berinjela padrão. Para não aumentar muito o número de classes de alimentos, é necessário criar uma que englobe todos esses alimentos "diferentes". Portanto, ter uma classe que não possui características bem definidas dificulta a aprendizagem da rede.

A fim de verificar todo o potencial dos dados de treinamento, foi treinada uma rede com uma divisão mais simples dos dados, apenas 80% de treinamento e 20% de teste. O resultados foram bem similares à média da validação cruzada, com 50% de acurácia e 17,35% mIOU. Os gráficos da figura 5.2, extraídos da ferramenta WandB [66], mostram como foi a evolução das métricas ao longo do treinamento.

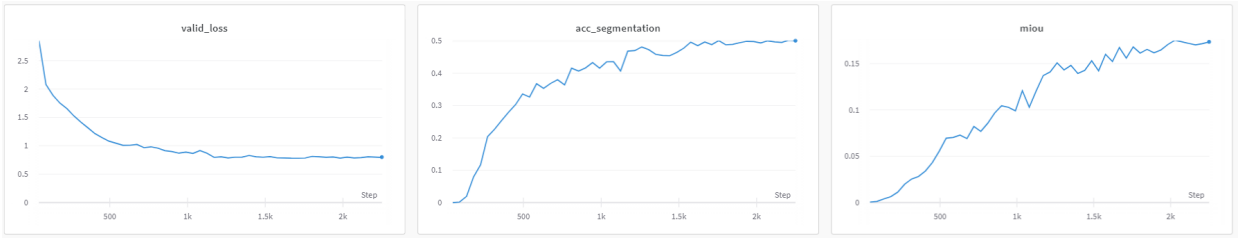


Figura 5.2: *Valid loss* representa a função de custo para os dados de teste, *acc segmentation* representa acurácia e *miou* representa mIOU.

Conforme já mencionado, a métrica mIOU foi muito baixa e significa que o modelo não aprendeu a classificar e segmentar alimentos, dada à dificuldade inerente de classificar alimentos e à complexidade do banco de dados (poucas imagens e muitas classes). A figura 5.3 confirma a hipótese, mostrando as predições da rede para diversas imagens do *dataset* de teste. Nota-se a falta de consistência ao detectar diversas classes de alimentos, principalmente aquelas que aparecem menos. Porém, para imagens mais simples, com apenas um alimento, percebe-se que foi possível segmentar com precisão (primeira imagem com macarrão). Apesar da dificuldade de classificar os alimentos, o algoritmo foi capaz de diferenciar o que é alimento e o que não é.

Figura 5.3: Imagens à esquerda são as fotos originais do *dataset* de teste e imagens à direita são as predições da rede neural.

5.2.2 Sprint 1

Tendo em vista as hipóteses mencionados para a baixa performance do modelo no Sprint 0, uma possível abordagem é reduzir o número de classes de alimentos, já que o banco não tem imagens suficientes para que a rede aprenda sobre todas elas. Conforme já explicado na subseção 4.1.4.2, Sprint 1 agrupou as classes de alimentos em 16 grandes grupos. Mantendo o mesmo algoritmo de segmentação e de validação cruzada, obteve-se os resultados da tabela 5.6.

Tabela 5.6: Validação cruzada para os dados do Sprint 1.

Cross-Validation Fold	Dataset	Acurácia	mIOU
1	Validação	69,1%	37%
	Teste	65,8%	33,7%
2	Validação	68,4%	34,6%
	Teste	67,1%	35,4%
3	Validação	63,8%	32,7%
	Teste	64,1%	33,5%
4	Validação	64,3%	29,5%
	Teste	68,4%	36,7%
5	Validação	64,3%	33,9%
	Teste	65,9%	33,8%
Média	Validação	66%	33,5%
	Teste	66,3%	34,6%

Nota-se uma melhoria considerável no resultado do mIOU médio (igual a 34,6%), em comparação com os 16,3% do Sprint 0. Isso mostra de forma objetiva como a granularidade das classes do banco de dados prejudica a performance, ainda mais para uma tarefa tão difícil como segmentação de alimentos, em que é necessário uma grande variedade de imagens representando o mesmo alimento, para que assim o modelo aprenda todas as suas características e diferentes modos de preparo.

A fim de verificar todo o potencial dos dados de treinamento, foi treinada uma rede com uma divisão mais simples dos dados, apenas 80% de treinamento e 20% de teste. O resultados foram bem melhores em relação à média da validação cruzada, com 73% de acurácia e 45% mIOU. Os gráficos da figura 5.4 mostram como foi a evolução das métricas ao longo do treinamento.

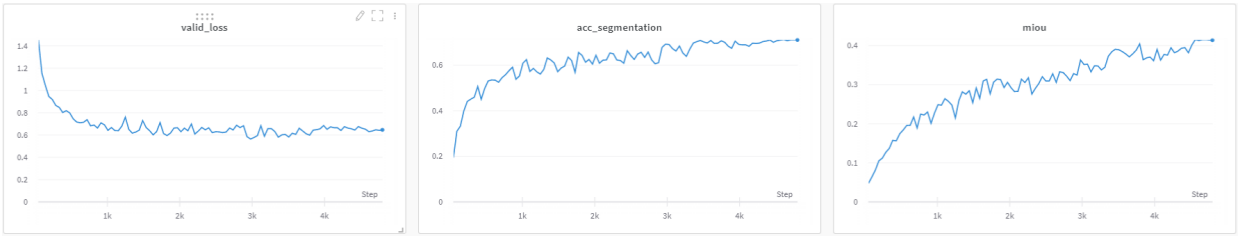


Figura 5.4: *Valid loss* representa a função de custo para os dados de teste, *acc segmentation* representa acurácia e *miou* representa mIOU.

Com um melhor resultado de segmentação semântica, já é possível ver que a rede é capaz de identificar corretamente alguns pratos de alimento, conforme mostra a figura 5.5. Mesmo que seja uma tarefa mais simples, onde são identificados apenas grupos de alimentos ao invés do alimento em si, já é possível expandir isso para aplicações que avaliam a dieta do brasileiro: pratos com os

macronutrientes desbalanceados ou falta de legumes podem indicar que é necessário intervir e ajudar este paciente. Apesar do otimismo e de vários pratos de alimento terem sido identificados completamente, ainda seria necessário aumentar a base de dados e melhorar a métrica mIOU para colocar um sistema desse em produção.

Figura 5.5: Imagens à esquerda são as fotos originais do *dataset* de teste e imagens à direita são as predições da rede neural.

Uma outra forma de analisar os resultados obtidos desse Sprint é entender como foi a performance de cada classe. A tabela 5.7 mostra a métrica f1-score para cada grupo e a média micro e macro final.

Tabela 5.7: Avaliação f1-score para cada grupo de alimento. A coluna *support* mostra quantas vezes determinado pixel de uma classe aparece nas imagens de teste.

	Precision	**Recall**	**F1-score**	**Support**
AlimentoNaoRegistrado	0.16	0.11	0.13	1035158
TiposArroz	0.78	0.74	0.76	676648
Leguminosas	0.76	0.77	0.76	736392
Frutas	0.53	0.33	0.41	35153
Legumes	0.71	0.68	0.69	1035158
Batatas	0.69	0.67	0.68	676648
Pure	0.83	0.48	0.60	736392
Salada	0.76	0.83	0.79	35153
CarneAnimal	0.74	0.83	0.78	1035158
Massa	0.81	0.56	0.66	676648
Queijos	0.08	0.38	0.13	736392
Farofa	0.37	0.53	0.44	35153
Ovo	0.85	0.63	0.73	1035158
Paes	0.65	0.49	0.56	676648
Torta	0.01	0.02	0.01	736392
Caldos	0.76	0.13	0.23	35153
Micro Avg	0.72	0.71	0.72	2953743
Macro Avg	0.59	0.51	0.52	2953743
Weighted Avg	0.72	0.71	0.71	2953743

Fica entendido pela análise das colunas f1-score e *support* que, quanto mais vezes a classe aparece no banco de dados, melhor é o resultado. Observe que a detecção de arroz, leguminosas, carne e salada é muito boa, pois esses alimentos aparecem muito. Já queijos, torta e caldos aparecem pouco e o modelo não aprendeu a detectá-los, prejudicando a média final. Isso mostra que é necessário trabalhar ainda mais para o balanceamento do banco, porém é uma tarefa particularmente difícil no contexto dos alimentos. Imagine, por exemplo, que todo prato que contém a classe "farofa"também vem acompanhada de "tiposArroz"e "leguminosas". Aumentar o número de imagens com "farofa"significa também aumentar as classes que já aparecem muito, implicando em manter o mesmo desbalanceamento. É necessário encontrar um número mínimo de vezes que a classe deve aparecer no banco para que o resultado seja de qualidade, mesmo que o desbalanceamento seja inerente nesse caso.

Outra classe que parece prejudicar a performance final do modelo é a classe "AlimentoNaoRegistrado", conforme já explicada na subseção 5.2.1 Sprint 0. Como o modelo fica confuso em identificá-la devido a falta de padrões e por abranger inúmeros alimentos diferentes, seu f1-score foi de apenas 13%, contribuindo para reduzir a média final.

Foram feitos alguns testes para verificar se a remoção das imagens com a classe "AlimentoNao-

Registrado"melhoraria o resultado. Porém esse tipo de teste vem com um custo, já que muitas imagens deveriam ser removidas da base. Apesar desse prejuízo, notou-se que os resultados foram melhores, com mIOU de 50%. Apesar da melhoria de 5%, outra vantagem foi identificada: a média macro do f1 score, que representa a qualidade do modelo ao classificar todas as classes possíveis, aumentou consideravelmente de 52% para 60%. Isso mostra que retirar a confusão causada por "AlimentoNaoRegistrado"melhorou o entendimento do modelo sobre os outros alimentos e diminuiu os desbalanceamento entre estes.

5.2.3 Sprint 2

Já sabendo do sucesso do experimento feito no Sprint 1, decidiu-se reduzir ainda mais a granularidade das classes para apenas 4 grandes grupos de alimentos: carboidratos, proteina, vegetais e frutas. Essa separação menos específica também poderia resultar em aplicações para avaliação de dietas, sendo possível identificar como a pessoa se alimenta e qual a proporção do seu prato para cada um dos macronutrientes predominantes nos grupos alimentares desse Sprint 2. Mantendo o mesmo algoritmo de segmentação e de validação cruzada, obteve-se os resultados da tabela 5.8.

Tabela 5.8: Validação cruzada para os dados do Sprint 2.

Cross-Validation Fold	**Dataset**	**Acurácia**	**mIOU**
1	Validação	80,4%	57,4%
	Teste	78,4%	57,2%
2	Validação	78,9%	60,7%
	Teste	79,1%	58,5%
3	Validação	75,3%	56,2%
	Teste	79,1%	58%
4	Validação	79,5%	63,9%
	Teste	80,9%	61,5%
5	Validação	81,1%	57,6%
	Teste	79,6%	58,4%
Média	Validação	79%	59,1%
	Teste	79,4%	58,7%

Novamente, reduzir o número de classes resultou em um mIOU muito maior, dessa vez chegando a 58,7% nos dados de teste. Isso já se torna uma métrica razoável para o problema, sendo este modelo capaz de identificar diversos pratos de alimento e os grupos de alimentos contidos neles. A figura 5.6 mostra os resultados para algumas imagens da base de teste.

Figura 5.6: Imagens à esquerda são as fotos originais do *dataset* de teste e imagens à direita são as predições da rede neural.

Novamente, para analisar o potencial máximo dos dados de treinamento, foi feita uma divisão mais simples em 80% de dados de treinamento e 20% de teste, ao invés de usar *cross-validation*. As métricas superaram a validação cruzada, com 84% de acurácia e 66% de mIOU, já que mais dados estavam disponíveis para o treinamento. A seguir, na figura 5.7, é possível ver essas métricas ao longo das épocas.

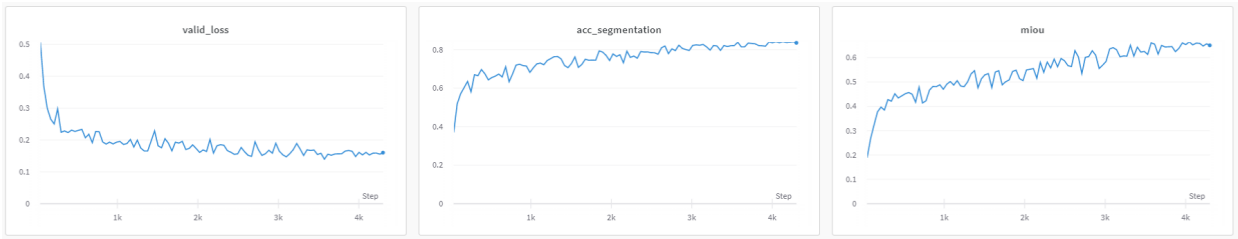


Figura 5.7: *Valid loss* representa a função de custo para os dados de teste, *acc segmentation* representa acurácia e *miou* representa mIOU.

A análise da performance de cada classe separadamente é interessante nesse caso também, para entender o que pode ser melhorado na basa de treinamento. Pela tabela 5.9, nota-se que a baixa quantidade de frutas prejudicou a média final do modelo, sendo necessário anotar mais imagens com esses alimentos.

Tabela 5.9: Avaliação f1-score para cada grupo de alimento. A coluna *support* mostra quantas vezes determinado pixel de uma classe aparece nas imagens de teste.

	Precision	**Recall**	**F1-score**	**Support**
Carbs	0.87	0.83	0.85	1035158
Proteina	0.81	0.82	0.82	676648
Vegetais	0.85	0.89	0.87	736392
Frutas	0.53	0.41	0.47	35153
Micro Avg	0.84	0.84	0.84	2483351
Macro Avg	0.77	0.74	0.75	2483351
Weighted Avg	0.84	0.84	0.84	2483351

Por fim, foi realizada uma última análise com os dados desse Sprint 2. Para justificar a escolha do algoritmo de segmentação semântica com UNET e *backbone* Resnet34, foram feitos vários testes mudando a arquitetura *backbone* responsável por codificar as características das imagens. A tabela 5.10 mostra a média da validação cruzada para os dados de validação e teste em cada arquitetura diferente. Além disso, esse mesmo resultado pode ser visto de forma gráfica nas figuras 5.8, 5.9 e 5.10.

Tabela 5.10: Média da Validação cruzada para diferentes arquiteturas e usando os dados de treinamento do Sprint 2.

Arquitetura	**Dataset**	**Acurácia**	**mIOU**
RESNET34	Validação	79%	59,1%
	Teste	**79,4%**	58,7%
RESNET101	Validação	75,7%	57,5%
	Teste	78,5%	**60,1%**
VGG-16	Validação	74,2%	54,8%
	Teste	76,1%	55,7%
AlexNET	Validação	68,4%	49%
	Teste	68,1%	45,9%

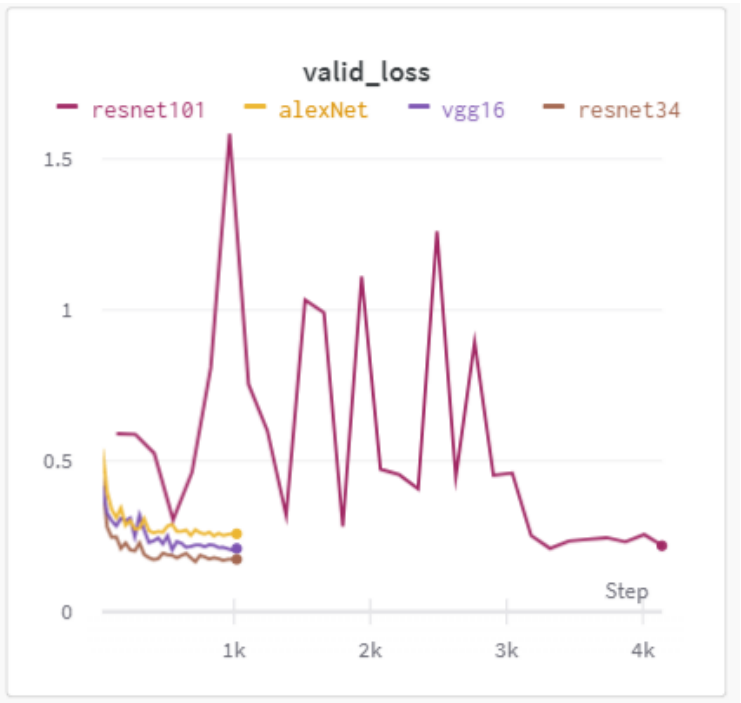


Figura 5.8: Comparação da função de custo para diferentes arquiteturas *backbone* do modelo Unet de segmentação semântica. Nota-se que Resnet34 teve bom desempenho e é umas das redes menos custosas computacionalmente. É curioso observar que redes maiores, como Resnet101, necessitam de muito mais épocas para convergirem.

Figura 5.9: Comparação da acurácia para diferentes arquiteturas *backbone* do modelo Unet de segmentação semântica.

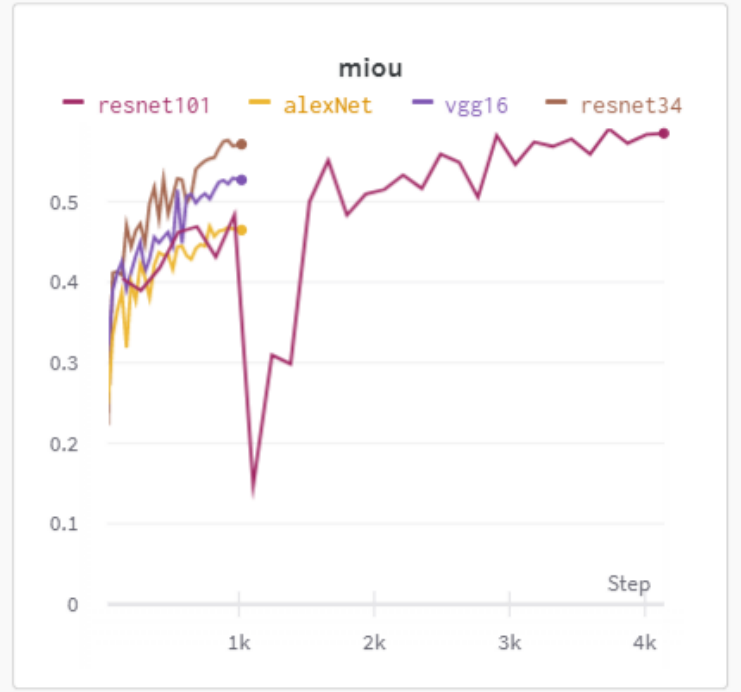


Figura 5.10: Comparação do mIOU para diferentes arquiteturas *backbone* do modelo Unet de segmentação semântica.

Apesar de inúmeras possibilidades de arquitetura hoje em dia, Resnet34 mostrou ter um bom compromisso entre custo computacional e desempenho. Por ser uma arquitetura extremamente eficiente, resultou na melhor acurácia para os dados de teste e o segundo melhor mIOU, atrás apenas do seu similar Resnet101 (mesma arquitetura, porém maior e computacionalmente mais pesada). Isso justifica porque foi escolhido esse *backbone* como padrão no algoritmo de segmentação implementado.

Capítulo 6

Conclusões

Dada toda a contextualização do problema e possíveis aplicações, ficou claro que segmentação semântica de alimentos é uma área muito interessante de se estudar e aperfeiçoar. Há várias formas de usar essa solução em aplicações diferentes na sociedade, entre elas avaliação de dieta, ajuda no emagrecimento e auxílio para diabéticos contarem carboidratos em suas refeições. Apesar de existirem vários estudos no mundo inteiro, no Brasil ainda é muito escasso, sendo necessário explorar ainda mais as nuances da culinária brasileira e criar base de dados robustas.

Com isso em mente, foi criado nesse estudo uma base de imagens que representa o cotidiano do brasileiro, sem se preocupar em buscar imagens em ambientes controlados. Para isso, também foi desenvolvida uma ferramenta que ajuda na anotação e criação dos fundos verdades de cada foto. Com o banco de imagens em mãos, foi necessário explorar a literatura e encontrar modelos de aprendizagem profunda que fossem capaz de segmentar e classificar objetos em uma imagem (nesse caso, alimentos), como o modelo Unet e o *backbone* Resnet34.

Foram feitos vários testes desse modelo escolhido, usando um algoritmo de segmentação fixo. Primeiro avaliou-se a qualidade do modelo em *datasets* públicos, em que foi possível comparar a performance do nosso modelo com o resultado dos artigos que criaram essas bases. Essa etapa foi extremamente positiva, já que nosso trabalho conseguiu superar as métricas de 3 de 4 bases públicas testadas na tarefa de segmentação de alimentos.

Após a confirmação que o algoritmo implementado era robusto, foram feitos 3 grandes testes na base de alimentos criada com alimentos brasileiras, sendo cada teste nomeado como um Sprint diferente. Primeiro, no Sprint 0, cada alimento era representado por uma classe, e foi observado um baixo desempenho devido ao grande número de classes e baixo número de imagens no *dataset*. Após esse teste, no Sprint 1, agrupou-se alimentos parecidos em grupos maiores, reduzindo, portanto, a granularidade do banco para apenas 16 classes distintas de alimentos. Isso fez que a performance aumentasse consideravelmente, mostrando que é necessário coleta mais imagens para criar um modelo robusto que é capaz de identificar e classificar cada alimento separadamente (grau máximo de granularidade). No Sprint 2, reduziu-se ainda mais o número de classes, em apenas 4 grandes grupos alimentícios. O resultado mostrou o esperado: o desempenho melhorou ainda mais. Além dos testes mencionados, foi feito um comparativo para justificar

a escolha do modelo Unet com *backbone* Resnet34. Usando os dados do Sprint 2, verificou-se que Resnet34 representa um bom compromisso entre custo computacional e desempenho, comparado a Resnet101, VGG16 e AlexNet.

Por fim, conclui-se que foi aprendido as melhores práticas no desenvolvimento de redes neurais com aprendizagem profunda. Técnicas como *Data Augmentation*, *Focal Loss*, *Cross-Validation* e *Fit-one-cycle* foram usadas no algoritmo de segmentação e ajudaram na obtenção do resultados expressivos. Além disso, foi necessário entender a fundo o funcionamento de bibliotecas de *Deep Learning*, como Fastai e Pytorch, contribuindo bastante para o aprendizado prático da disciplina. Outro grande ganho desse projeto foi entender como a qualidade dos dados afeta os resultados do modelo. Independente se é usado uma boa ferramenta, realizar uma tarefa em que os dados de treinamento são mal anotados ou inconsistentes se torna muito difícil.

6.1 Perspectivas Futuras

Apesar de bons resultados terem sido obtidos nos Sprints 1 e 2, ainda há muito espaço para melhoria, principalmente para o caso do Sprint 0, em que cada alimento é representado por uma classe. Há três próximos passos: aumentar o banco de dados, organizá-lo melhor (eliminando imagens complexas e confusas) e retirando anotações inconsistentes do banco. Sugere-se focar em estratégias focadas na melhoria dos dados, mantendo o algoritmo de segmentação fixo, conforme sugerido pela abordagem *Data-Centric*.

Ainda que as métricas mIOU obtidas nos Sprints 0, 1 e 2 não tenham sido altos, eles foram condizentes com outros estudos que realizaram segmentação semântica de alimentos. Inclusive, no estudo FoodSeg103 [3] é mostrado um comparativo do mIOU com a tarefa de segmentação semântica de objetos de cena (poste, ruas, edifícios), mostrando como é desafiador encontrar modelos capazes de segmentar alimentos de forma robusta (ver Figura 6.1). Isso mostra que há um grande espaço para explorar essa área, lembrando que ainda é necessário ter bases de dados específicas de cada culinária para criar aplicações relevantes.

Methods	Cityscapes	FoodSeg103	*gap*
CCNet	79.0	35.0	34.0
Sem-FPN	74.5	27.8	46.7
SeTR	77.9	41.3	36.6

Table 4: Semantic segmentation results on Cityscape [8] and our FoodSeg103, showing that our FoodSeg103 is much more challenging than the object image dataset for the task of semantic segmentation.

Figura 6.1: Comparação do mIOU entre segmentação semântica de alimentos e objetos de cena. Nota-se na última coluna a diferença expressiva de resultados e como é desafiadora essa tarefa de identificação de alimentos [3].

Além das melhorias na segmentação semântica, ainda é preciso trabalhar em técnicas para estimar a quantidade de nutrientes a partir de uma imagem. Isso envolve estudar disciplinas como reconstrução 3D, oclusão, relação entre o peso do prato e a imagem, obtenção de informação a

partir de vídeos e impacto social de tal aplicação.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MIN, W. et al. *A Survey on Food Computing*. 2019.

[2] BENNY. *Yolo for Real Time Food Detection*. 2018. Disponível em: <http://bennycheung.github.io/yolo-for-real-time-food-detection>.

[3] WU, X. et al. A large-scale benchmark for food image segmentation. In: *Proceedings of ACM international conference on Multimedia*. [S.l.: s.n.], 2021.

[4] MARíN, J. et al. Recipe1m: A dataset for learning cross-modal embeddings for cooking recipes and food images. 10 2018.

[5] CARVALHO, B.; BORGES, G. Segmentação de alimentos e avaliação de uma ferramenta para estimação da quantidade de carboidrato. *PIBIT*, p. 8, 10 2019.

[6] ROSEBROCK, A. *WaterShed OpenCV*. 2015. Disponível em: <https://www.pyimagesearch.com/2015/11/02/watershed-opencv/>.

[7] FASTAI. *FastAI Documentation*. 2019. Disponível em: <https://docs.fast.ai/>.

[8] SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks*. 2018. Disponível em: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

[9] MUJTABA, H. *Resnet Explained*. 2020. Disponível em: <https://www.mygreatlearning.com/blog/resnet>.

[10] RONNEBERGER, O.; FISCHER, P.; BROX, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015.

[11] HOSTENS, E. Quantum entanglement distillation in the stabilizer formalism. 10 2021.

[12] LIN, T.-Y. et al. *Focal Loss for Dense Object Detection*. 2018.

[13] KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017.

[14] RABELLO, E. B. *Cross Validation: Avaliando seu modelo de Machine Learning*. 2019. Disponível em: <https://medium.com/@edubrazrabello/cross-validation-avaliando-seu-modelo-de-machine-learning-1fb70df15b78>.

[15] TIU, E. *Metrics to Evaluate your Semantic Segmentation Model*. 2019. Disponível em: <https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>.

[16] CIOCCA, G.; NAPOLETANO, P.; SCHETTINI, R. Food recognition: a new dataset, experiments and results. *IEEE Journal of Biomedical and Health Informatics*, IEEE, v. 21, n. 3, p. 588–598, 2017.

[17] POULADZADEH, P.; SHIRMOHAMMADI, S.; ALMAGHRABI, R. Measuring calorie and nutrition from food image. *Instrumentation and Measurement, IEEE Transactions on*, v. 63, p. 1947–1956, 08 2014.

[18] GLIC. *Aplicativo para diabetes e controle de glicemia*. 2021. Disponível em: <https://gliconline.net/>.

[19] FOODVISOR. *Aplicativo de nutrição*. 2021. Disponível em: <https://www.foodvisor.io/en/>.

[20] OKAMOTO, K.; YANAI, K. UEC-FoodPIX Complete: A large-scale food image segmentation dataset. In: *Proc. of ICPR Workshop on Multimedia Assisted Dietary Management(MADiMa)*. [S.l.: s.n.], 2021.

[21] FREITAS, C. N. C.; CORDEIRO, F. R.; MACARIO, V. *MyFood: A Food Segmentation and Classification System to Aid Nutritional Monitoring*. 2020.

[22] ANNOTATION, S. *Super Annotation*. 2021. Disponível em: <https://superannotate.com/>.

[23] MIN, W. et al. *ISIA Food-500: A Dataset for Large-Scale Food Recognition via Stacked Global-Local Attention Network*. 2020.

[24] CARVALHO, T. Q. B. *Dataset de comidas brasileiras BR-UMAS21*. 2021. Disponível em: <https://bit.ly/brumas21>.

[25] ELSWEILER, D.; TRATTNER, C.; HARVEY, M. Exploiting food choice biases for healthier recipe recommendation. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2017. (SIGIR '17), p. 575–584. ISBN 978-1-4503-5022-8. Disponível em: <http://doi.acm.org/10.1145/3077136.3080826>.

[26] HASSANNEJAD, H. et al. Food image recognition using very deep convolutional networks. In: *Proceedings of the 2Nd International Workshop on Multimedia Assisted Dietary Management*. New York, NY, USA: ACM, 2016. (MADiMa '16), p. 41–49. ISBN 978-1-4503-4520-0. Disponível em: <http://doi.acm.org/10.1145/2986035.2986042>.

[27] ANTHIMOPOULOS, M. et al. Segmentation and recognition of multi-food meal images for carbohydrate counting. In: IEEE. *13th IEEE International Conference on BioInformatics and BioEngineering*. [S.l.], 2013. p. 1–4.

[28] LIANG, Y.; LI, J. Computer vision-based food calorie estimation: dataset, method, and experiment. *CoRR*, abs/1705.07632, 2017. Disponível em: <http://arxiv.org/abs/1705.07632>.

[29] POULADZADEH, P. et al. Mobile cloud based food calorie measurement. In: *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. [S.l.: s.n.], 2014. p. 1–6.

[30] Myers, A. et al. Im2calories: Towards an automated mobile vision food diary. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2015. p. 1233–1241.

[31] AHN, Y.-Y. et al. Flavor network and the principles of food pairing. *Scientific Reports*, v. 1, 11 2011.

[32] LU, Y.; HUANG, Y.; LU, R. Innovative hyperspectral imaging-based techniques for quality evaluation of fruits and vegetables: A review. *Applied Sciences*, v. 7, n. 2, 2017. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/7/2/189>.

[33] PABICO, J. P.; GRANO, A. V. D.; ZARSUELA, A. L. *Neural Network Classifiers for Natural Food Products*. 2015.

[34] PATEL, V. C.; MCCLENDON, R.; GOODRUM, J. Color computer vision and artificial neural networks for the detection of defects in poultry eggs. *Artificial Intelligence Review*, v. 12, p. 163–176, 2004.

[35] OFLI, F. et al. Is saki delicious?: The food perception gap on instagram and its relation to health. In: . [S.l.: s.n.], 2017. p. 509–518.

[36] SRIVASTAVA, N.; SALAKHUTDINOV, R. Multimodal learning with deep boltzmann machines. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*. Red Hook, NY, USA: Curran Associates Inc., 2012. (NIPS'12), p. 2222–2230.

[37] BOSSARD, L.; GUILLAUMIN, M.; GOOL, L. V. Food-101–mining discriminative components with random forests. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2014. p. 446–461.

[38] DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255.

[39] HE, K. et al. *Deep Residual Learning for Image Recognition*. 2015.

[40] SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015.

[41] KAWANO, Y.; YANAI, K. Food image recognition with deep convolutional features. In: *Proc. of ACM UbiComp Workshop on Cooking and Eating Activities (CEA)*. [S.l.: s.n.], 2014.

[42] REDMON, J. et al. You only look once: Unified, real-time object detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 779–788.

[43] REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA: MIT Press, 2015. (NIPS'15), p. 91–99.

[44] CHEN, L.-C. et al. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017.

[45] GE, M.; RICCI, F.; MASSIMO, D. Health-aware food recommender system. In: *Proceedings of the 9th ACM Conference on Recommender Systems*. New York, NY, USA: Association for

Computing Machinery, 2015. (RecSys '15), p. 333–334. ISBN 9781450336925. Disponível em: <https://doi.org/10.1145/2792838.2796554>.

[46] FORSYTH; PONCE. *Computer Vision - A Modern Approach*. 1. ed. [S.l.]: Pearson.

[47] LIBRARY, D. P. *Difference Hashing*. 2021. Disponível em: <https://pypi.org/project/dhash/>.

[48] MASTERY, M. L. *A Gentle Introduction to Information Entropy*. 2021. Disponível em: <https://machinelearningmastery.com/what-is-information-entropy/>.

[49] SMITH, L. N.; TOPIN, N. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017. Disponível em: <http://arxiv.org/abs/1708.07120>.

[50] KAWANO, Y.; YANAI, K. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In: *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*. [S.l.: s.n.], 2014.

[51] CIOCCA, G.; NAPOLETANO, P.; SCHETTINI, R. Learning cnn-based features for retrieval of food images. In: BATTIATO, S. et al. (Ed.). *New Trends in Image Analysis and Processing – ICIAP 2017: ICIAP International Workshops, WBICV, SSPandBE, 3AS, RGBD, NIVAR, IWBAAS, and MADiMa 2017, Catania, Italy, September 11-15, 2017, Revised Selected Papers*. [S.l.]: Springer International Publishing, 2017. p. 426–434. ISBN 978-3-319-70742-6.

[52] CIOCCA, G.; NAPOLETANO, P.; SCHETTINI, R. Cnn-based features for retrieval and classification of food images. *Computer Vision and Image Understanding*, Elsevier, -, p. –, 2018.

[53] MARTINEL, N.; FORESTI, G. L.; MICHELONI, C. *Wide-Slice Residual Networks for Food Recognition*. 2016.

[54] UNIMIB. *Benchmarking Algorithms for Food Localization and Semantic Segmentation*. 2021. Disponível em: <http://www.ivl.disco.unimib.it/activities/benchmarking-food-segmentation/>.

[55] PFISTERER, K. J. et al. *Fully-Automatic Semantic Segmentation for Food Intake Tracking in Long-Term Care Homes*. 2019.

[56] OKAMOTO, K.; YANAI, K. An automatic calorie estimation system of food images on a smartphone. In: *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*. New York, NY, USA: Association for Computing Machinery, 2016. (MADiMa '16), p. 63–70. ISBN 9781450345200. Disponível em: <https://doi.org/10.1145/2986035.2986040>.

[57] POULADZADEH, P.; SHIRMOHAMMADI, S. Mobile multi-food recognition using deep learning. *ACM Trans. Multimedia Comput. Commun. Appl.*, Association for Computing Machinery, New York, NY, USA, v. 13, n. 3s, ago. 2017. ISSN 1551-6857. Disponível em: <https://doi.org/10.1145/3063592>.

[58] UNITED States Department of Agriculture, USDA Food Composition Database. 2018. Disponível em: <https://ndb.nal.usda.gov/ndb/>.

[59] UNIVERSIDADE de São Paulo (USP), Tabela Brasileira de Composição de Alimentos (TBCA). 2018. Disponível em: <http://www.fcf.usp.br/tbca/>.

[60] ANDO, Y. et al. Depthcaloriecam: A mobile application for volume-based foodcalorie estimation using depth cameras. In: *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*. New York, NY, USA: Association for Computing Machinery, 2019. (MADiMa '19), p. 76–81. ISBN 9781450369169. Disponível em: <https://doi.org/10.1145/3347448.3357172>.

[61] Dehais, J. et al. Two-view 3d reconstruction for food volume estimation. *IEEE Transactions on Multimedia*, v. 19, n. 5, p. 1090–1099, 2017.

[62] MYFITNESSPAL. *Aplicativo de estilo de vida saudável*. 2021. Disponível em: <https://www.myfitnesspal.com/pt>.

[63] CALORIEMAMA. *Aplicativo para estimação de nutrientes a partir de uma foto*. 2021. Disponível em: <https://caloriemama.ai/>.

[64] CARVALHO, T. Q. B. *Ferramenta de anotação com Watershed e Notebooks de Segmentação Semântica de Alimentos Brasileiros*. 2021. Disponível em: <https://github.com/Bunoviske/diabeteslearningDeeplearning>.

[65] BARBOSA, L. *Feijão com arroz e arroz com feijão: o Brasil no prato dos brasileiros.* 2007. Disponível em: <https://doi.org/10.1590/S0104-71832007000200005>.

[66] WANDB. *Weights and Biases*. 2021. Disponível em: <https://wandb.ai/site>.

ANEXOS

I. PROGRAMAS UTILIZADOS

DiabetesLearning_BrazilFoods

29 de outubro de 2021

I.1 DIABETES LEARNING

```
[ ]: isGoogleColab = True
```

I.2 Notes

- check these blog posts: https://muellerzr.github.io/fastblog/ and his course: https://walkwithfastai.com/intro.contribute
- learner.export() already save all the transforms that will be applied during inference time, so it is not necessary to resize or normalize stats. If running outside the model outside fastai, check: https://forums.fast.ai/t/do-we-need-to-normalize-single-image-before-running-predict-function-on-it/44301/4
- to_fp16() is mixed precision
- item_tfms already apply the resize before collating the images in a batch. So it is not necessary to do this manually
- aug_tfms will only be applied on train_dl. validation and test will not be affected
- test_dl is a testing dataloader that uses the same transforms as train_dl and valid_dl, but with new data
- check images cleaner in the future
- cross validation: check walkwithfastai video 3 or https://forums.fast.ai/t/is-it-possible-to-implement-cross-validation-in-fastai/44961/15 or https://forums.fast.ai/t/am-i-doing-k-fold-cross-validation-right/84738/6

- stratification in Kfold and Train_Test split for Multi Label problems is not straighforward. Check http://scikit.ml/index.html#. Use for now balanced datasets!!!!

I.3 Installation

I.3.1 Kill all other GPU sessions

Run the first time this cell. Then, restart your session and run again the notebook without this code.

```
[ ]: #!kill -9 -1
```

I.3.2 Keep Session Alive

Put this javascript code in the browser console

```
[ ]: # function ClickConnect(){
#     console.log("Working");
#     document.querySelector("#top-toolbar > colab-connect-button").shadowRoot.
 ↪querySelector("#connect").click()
```

```
# }  setInterval(ClickConnect,300000)
```

I.3.3 Update ipython/ipykernel (Colab)

The Colab session can fail when running the first time this cell. Just run it a second time.

[ ]:
```
# This magic cell should be put first in your colab notebook.
# It'll automatically upgrade colab's really antique ipython/ipykernel to␣
 ↪their
# latest versions which are required for packages like ipyexperiments

# from packaging import version
# import IPython, ipykernel
# IPython.__version__
# if version.parse(IPython.__version__) <= version.parse("5.5.0"):
#     !pip install -q --upgrade ipython
#     !pip install -q --upgrade ipykernel

#     import os
#     import signal
#     os.kill(os.getpid(), signal.SIGTERM)
# print(f"ipykernel=={ipykernel.__version__}")
# print(f"IPython=={IPython.__version__}")
```

[ ]:
```
# new (large)
# !pip install ipyexperiments
# new (large)
# from ipyexperiments import IPyExperimentsPytorch
# exp = IPyExperimentsPytorch()
```

I.3.4 Check GPU e CPU RAM

[ ]:
```
if isGoogleColab:

    import torch
    from pynvml import *
    nvmlInit()
    def log_mem():
        h = nvmlDeviceGetHandleByIndex(0)
        info = nvmlDeviceGetMemoryInfo(h)
        print(f'GPU total Memory    : {info.total}')
        print(f'GPU free Memory     : {info.free}')
```

```
        print(f'GPU Memory used     : {info.used}')
    torch.ones(1).to(0)
    log_mem()

    !nvidia-smi
```

```
GPU total Memory    : 17071734784
GPU free Memory     : 16223305728
GPU Memory used     : 848429056
Thu Jun  3 21:09:08 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.27       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   51C    P0    39W / 250W |    809MiB / 16280MiB |      1%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

I.3.5 Mount the Google Drive to Google Colab

[ ]:
```
if isGoogleColab:
    from google.colab import drive
    drive.mount('/content/drive', force_remount=True)
    path_to_drive = "/content/drive/My Drive/Colab Notebooks/"
    sys.path.append(path_to_drive + 'DiabetesLearning/') #make src files␣
 ↪importable

else:
    path_to_drive = "./"
```

```
Mounted at /content/drive
```

[ ]:
```
#!unrar x "/content/drive/My Drive/Colab Notebooks/DiabetesLearning/
 ↪augmentedDataset_v2.rar" "/content/drive/My Drive/Colab Notebooks/
 ↪DiabetesLearning/"
#!unrar x "/content/drive/My Drive/Colab Notebooks/DiabetesLearning/
 ↪augmentedDataset.rar" # extrai local
#!unzip "/content/drive/My Drive/Colab Notebooks/DiabetesLearning/database.
 ↪zip" -d "/content/drive/My Drive/Colab Notebooks/DiabetesLearning/"
```

I.3.6 Install Deep Learning libraries

[ ]:
```
# !pip install torch==1.7.1
# !pip install fastai==2.3.1
# !pip install wandb

##restart runtime!!!
```

[ ]:
```
import sys; print('Python:',sys.version)
import torch; print('Pytorch:',torch.__version__)
import fastai; print('Fastai:',fastai.__version__)
```

```
Python: 3.7.10 (default, May  3 2021, 02:48:31)
[GCC 7.5.0]
Pytorch: 1.7.1
Fastai: 2.3.1
```

I.4 Train your model

[ ]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
from fastai.basics import *
from fastai.callback.all import *
from fastai.vision.all import *
from fastai.callback.wandb import *
import torch
import torchvision
from PIL import Image
import wandb
wandb.login()
```

```
wandb: Currently logged in as: bunoviske (use `wandb login
--relogin` to force relogin)
```

[ ]: True

I.4.1 Data

[ ]:
```
path = path_to_drive + 'DiabetesLearning/dataset_v1/sprint5/'
path_anno = path + 'gt/'
path_img = path + 'done/'

#funcao que pega a imagem de anotacoes correspondente, dado a imagem original␣
 ↪de entrada
# get_y_fn = lambda x : path_anno + '/' + f'{x.stem}_GT.png'
```

[ ]:
```
np.random.seed(2) #mesma semente para todas as vezes que executar
randomSeed = 2

label_fnames = get_image_files(path_anno)
print(label_fnames[:3])
fnames = get_image_files(path_img)
print(fnames[:3])

len(fnames), len(label_fnames)
```

```
[Path('/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/gt/570_GT.png'),
Path('/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/gt/562_GT.png'),
Path('/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/gt/136_GT.png')]
[Path('/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/158.jpg'),
Path('/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/547.jpeg'),
Path('/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/168.jpg')]
```

[ ]: (431, 431)

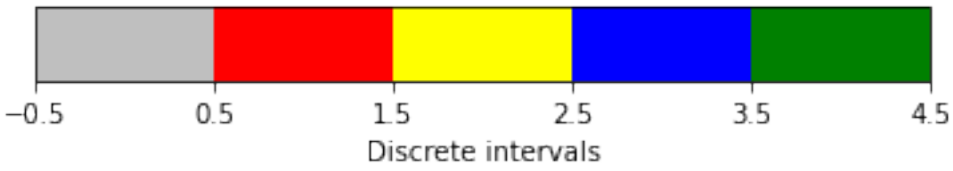
[ ]:
```
codes = np.loadtxt(path + 'classesNumber.txt', dtype=str,␣
 ↪delimiter='\n',encoding='utf')
codesId = [code.split(": ")[0] for code in codes]
codes = [code.split(": ")[1] for code in codes] #pega apenas o nome de cada␣
 ↪classe e ignora o ID
len(codes)
```

[ ]: 5

[ ]:
```
### fix GT ids: masks must be [0, 1, ..., K-1] where K is the number of␣
 ↪categories

def createAdjustedGroundTruthIds(fnames):
  "Gather the codes from a list of `fnames`"
  vals = set()
  for fname in fnames:
    msk = np.array(PILMask.create(fname))
    for val in np.unique(msk):
      if val not in vals:
        vals.add(val)
  vals = list(sorted(vals))

  for i,val in enumerate(vals):
    adjustedGroundTruthIds[i] = vals[i]

def get_y_fn(item):
  "Grab a mask from a `filename` and adjust the pixels based on␣
 ↪`adjustedGroundTruthIds`"
  fn = path_anno + '/' + f'{item.stem}_GT.png'
  msk = np.array(PILMask.create(fn))
  mx = np.max(msk)
  for i, val in enumerate(adjustedGroundTruthIds):
    msk[msk==adjustedGroundTruthIds[i]] = val
  return PILMask.create(msk)

adjustedGroundTruthIds = dict()
adjustedGroundTruthIds = {i : int(codesId[i]) for i in range(len(codes))}
# createAdjustedGroundTruthIds(label_fnames)
```

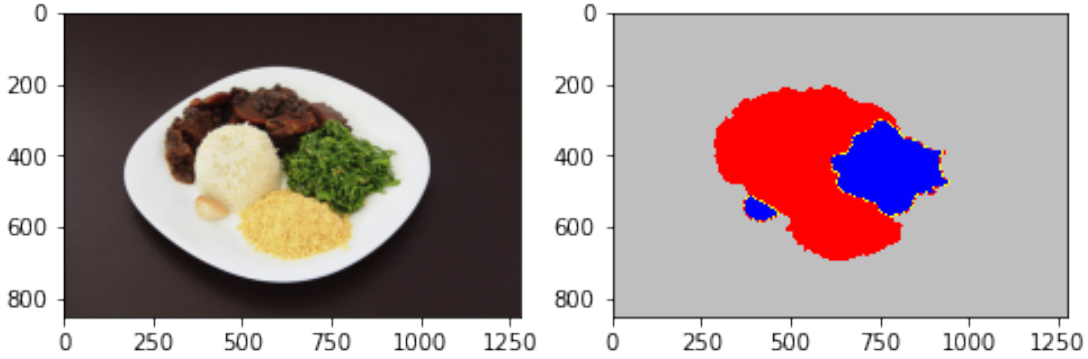
[ ]:
```
# define colormap for masks

fig, ax = plt.subplots(figsize=(6, 1))
fig.subplots_adjust(bottom=0.5)

cmap = matplotlib.colors.ListedColormap([np.random.rand(3,) for i in␣
 ↪range(0,len(codes))])
cmaplist = [cmap(i) for i in range(cmap.N)]
cmaplist[0] = (.75, .75, .75, 1.0) # nao alimento -> cinza
cmaplist[1] = "red" # nao registrado
```

```
cmaplist[2] = "yellow" #arroz integral
cmaplist[3] = "blue" #arroz branco
cmaplist[4] = "green" #feijao

cmap = matplotlib.colors.LinearSegmentedColormap.from_list(
    'Custom cmap', cmaplist, cmap.N)
bounds = [i-0.5 for i in range(0,len(codes)+1)] # intervalos devem conter␣
 ↪apenas o numero inteiro, entao ir de 0.5 em 0.5
norm = matplotlib.colors.BoundaryNorm(bounds, cmap.N)

cb2 = matplotlib.colorbar.ColorbarBase(ax, cmap=cmap,
                                       norm=norm,
                                       spacing='proportional',
                                       orientation='horizontal')
cb2.set_label('Discrete intervals')
fig.show()
```

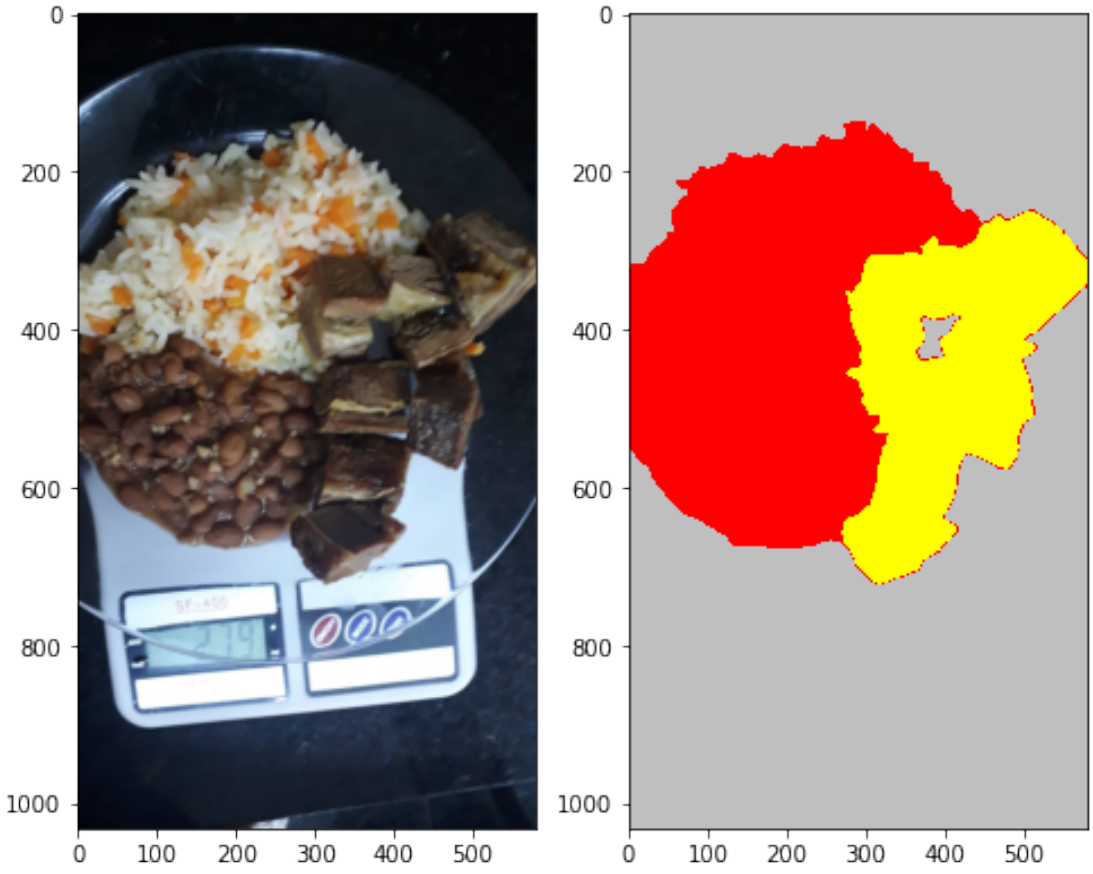


[ ]:
```
### VISUALIZACAO DA MÁSCARA FICA COM A BORDA ESTRANHA, MAS ELA ESTÁ CORRETA␣
 ↪COMO PODE SER VISTO NOS CODIGOS DO GROUND TRUTH

for idx in range(0,10):
  fig = plt.figure(figsize=(8,8))

  img = Image.open(fnames[idx])
  arr = np.asarray(img)
  ax1 = fig.add_subplot(1,2,1)
  ax1.imshow(arr)

  img1 = get_y_fn(fnames[idx]) # use this when 'adjustedGroundTruthIds' is␣
 ↪necessary
  # img1 = Image.open(get_y_fn(fnames[idx]))
  arr = np.asarray(img1)
  ax1 = fig.add_subplot(1,2,2)
  ax1.imshow(arr, cmap=cmap, norm=norm)
```

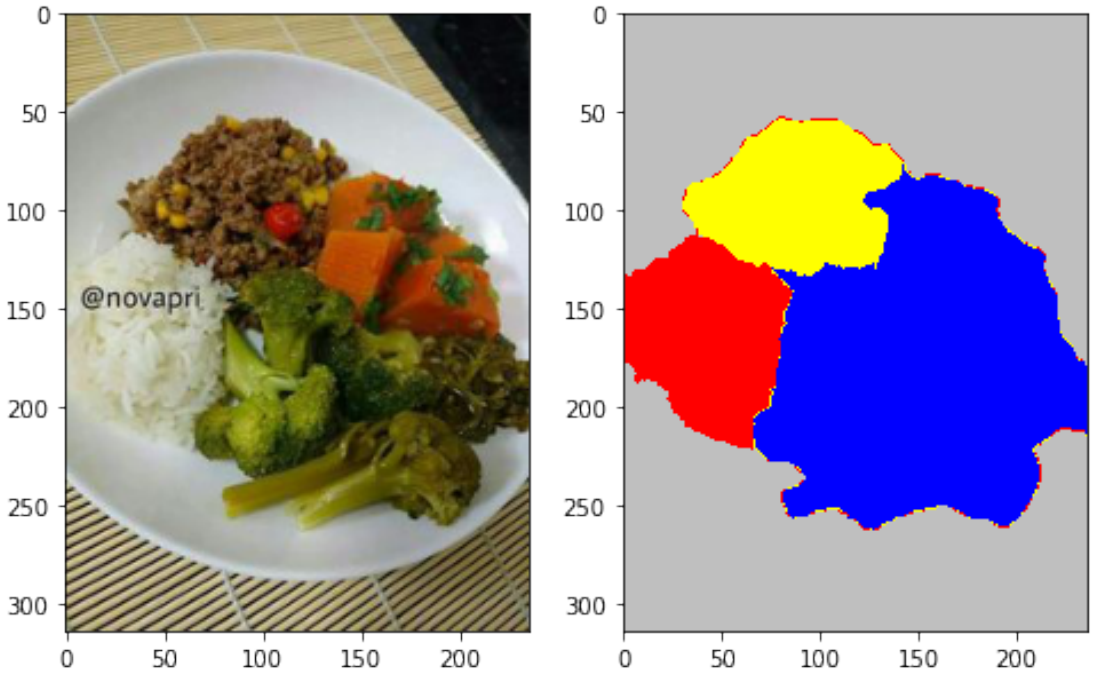
```
plt.show()

print("Ground truth Codes", np.unique(arr,return_counts=False))
print(fnames[idx])
```



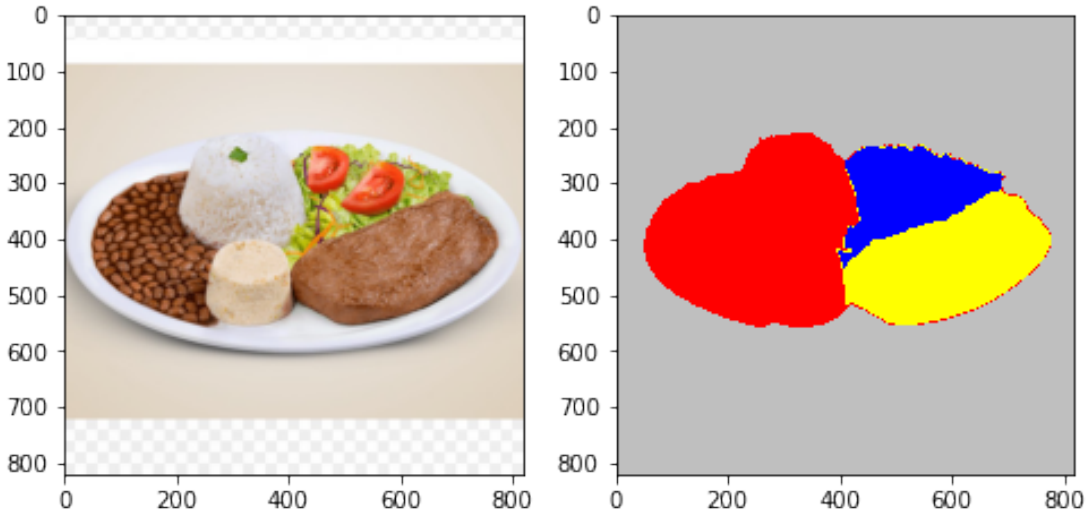

```
Ground truth Codes [0 1 3]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/158.jpg
```



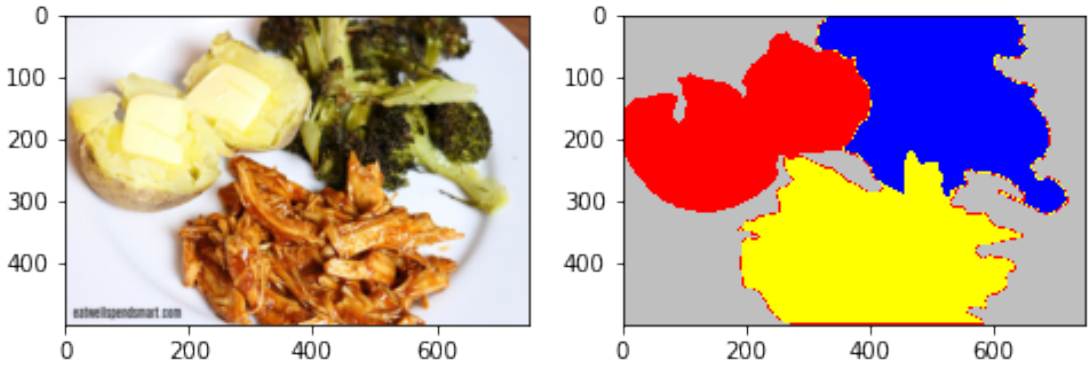
```
Ground truth Codes [0 1 2]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/547.jpeg
```



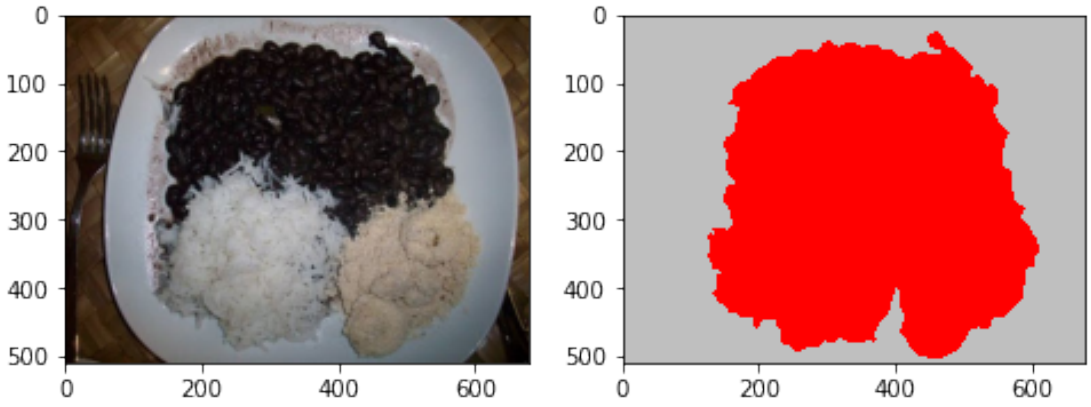

```
Ground truth Codes [0 1 2 3]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/168.jpg
```



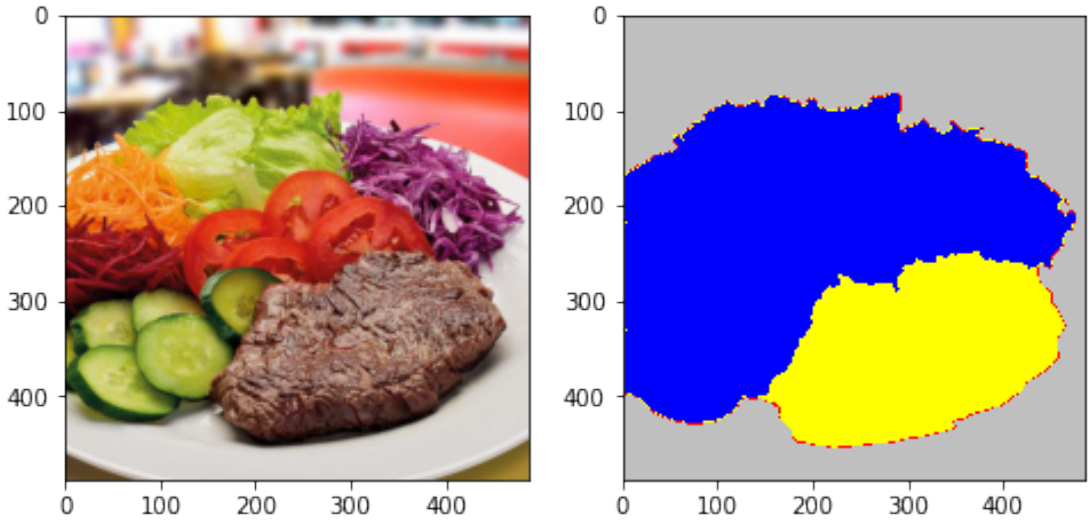
```
Ground truth Codes [0 1 2 3]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/154.jpg
```



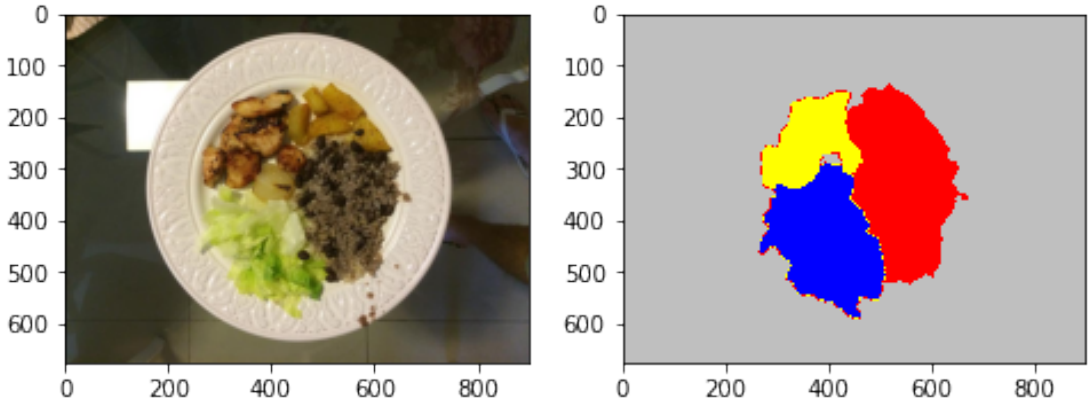

```
Ground truth Codes [0 1 2 3]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/1065.jpg
```



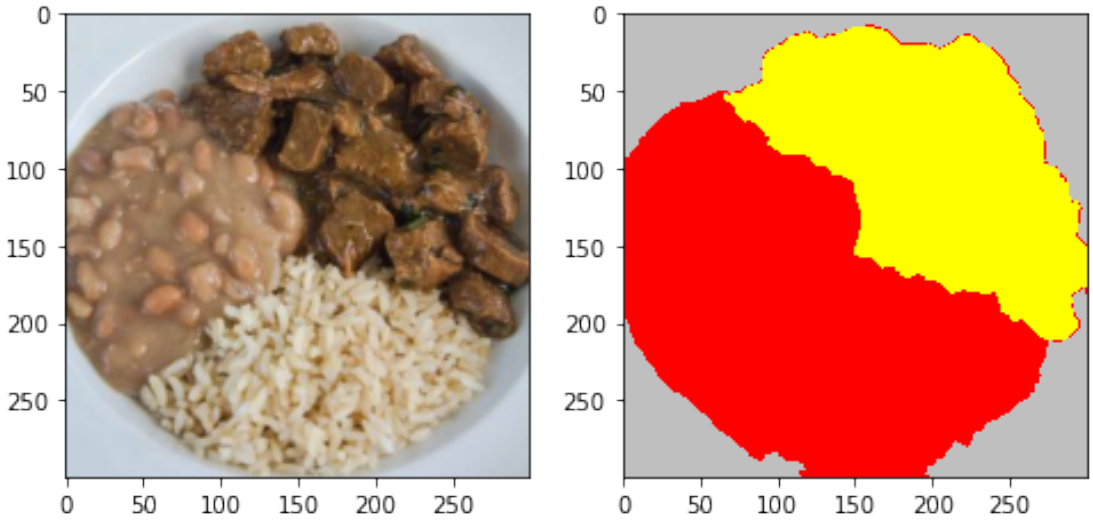

```
Ground truth Codes [0 1]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/892.jpg
```



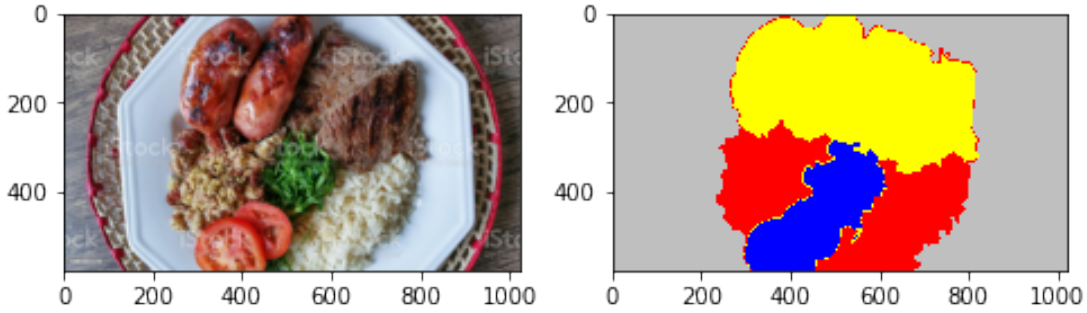

```
Ground truth Codes [0 2 3]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/190.jpg
```




```
Ground truth Codes [0 1 2 3]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/476.jpg
```




```
Ground truth Codes [0 1 2]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/165.jpg
```




```
Ground truth Codes [0 1 2 3]
/content/drive/My Drive/Colab
Notebooks/DiabetesLearning/dataset_v1/sprint5/done/856.jpg
```

I.4.2 Data Augmentation

[ ]:
```
size = (256, 256)
item_tfms = [Resize(size, method=ResizeMethod.Squish, resamples=(Image.
 ↪NEAREST,Image.NEAREST))]
aug_tfms = aug_transforms(mult=1, flip_vert=True, size=size)
```

I.4.3 Dataloaders

[ ]:
```
# dataset balance method. choose one of them or none

lossClassWeights = None
sampler = None

#### load class weights from pickle ####
# with open( path + "classWeightsSprint1.pkl", 'rb') as f:
#   weights = pickle.load(f)
# classWeights = torch.FloatTensor(weights).cuda()

#### assign class weights empirically ####
# classWeights = torch.ones(len(codes)).cuda()
# classWeights[0] = 0.1

### 1. class weights in loss function ###
# lossClassWeights = classWeights

### 2. oversampling ### not yet implemented, check here https://forums.fast.ai/
↪t/oversampling-in-fastai2/73721/14

# sampler = torch.utils.data.sampler.WeightedRandomSampler(classWeights,␣
↪len(classWeights))
# total_len_oversample = int(learn.data.c*np.max(label_counts))
# dataloaders.train_dl.dl.batch_sampler =␣
↪BatchSampler(WeightedRandomSampler(weights,total_len_oversample), data.
↪train_dl.batch_size,False)
```

[ ]:
```
# get data split from files or function

#### split train/test randomly ####
#from sklearn.model_selection import train_test_split
#X_train, X_test, _, _ = train_test_split(fnames, label_fnames, test_size=0.2,␣
↪random_state=randomSeed, shuffle=True, stratify=None)

#### split by validation.txt ####
# testFiles = np.loadtxt(path + 'validation.txt', dtype=str,␣
↪delimiter='\n',encoding='utf')
# X_test = [file for file in fnames if file.name in testFiles]

#### split by filenames array from pickle ####
```

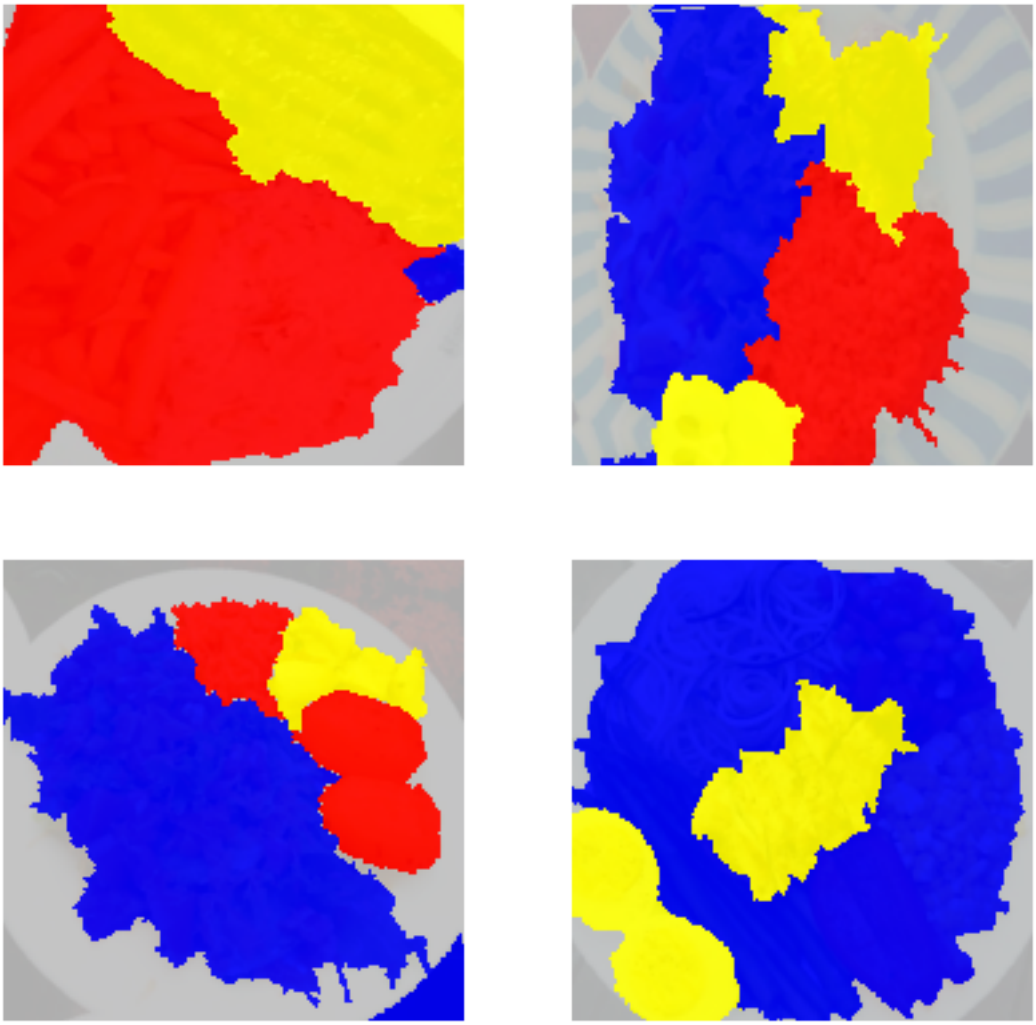
```
import pickle
with open(path + "testFilenamesSprint0.pkl", 'rb') as f:
  testFiles = pickle.load(f)
X_test = [file for file in fnames if file.name in testFiles]
```

[ ]:
```
bs = 8 # batch size
gradientAcc = int(32/bs) # 32 is the final "batch size"

trainBlock = DataBlock(blocks=(ImageBlock, MaskBlock(codes)),
                   get_items=get_image_files,
                   splitter=FuncSplitter(lambda o: o in X_test),
                   get_y=get_y_fn,
                   item_tfms=item_tfms,
                   batch_tfms=[*aug_tfms, Normalize.
 ↪from_stats(*imagenet_stats)])

dataloaders = trainBlock.dataloaders(path_img, path=path, bs=bs)

#### test dataloader ####
#### it is not working!!!!!!! test_dl always gives the same result

# testBlock = DataBlock(blocks=(ImageBlock, MaskBlock(codes)),
#                     get_items=get_image_files,
#                     splitter=FuncSplitter(lambda o: o in X_test),
#                     get_y=get_y_fn,
#                     item_tfms=item_tfms,
#                     batch_tfms=[Normalize.from_stats(*imagenet_stats)])
# testDataloaders = testBlock.dataloaders(path_img, path=path, bs=bs)

test_dl = dataloaders.test_dl(X_test, with_labels=True)
test_dl.vocab = codes
```

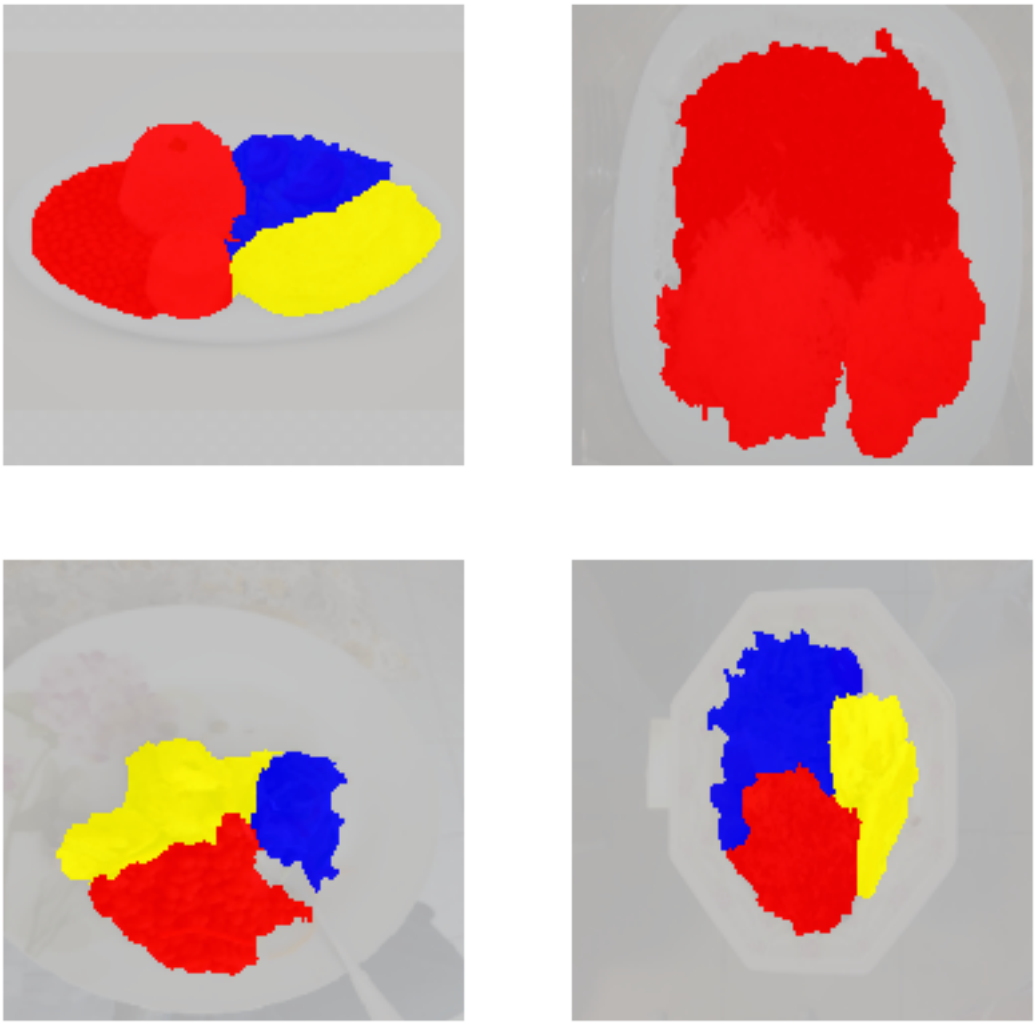
[ ]:
```
len(dataloaders.train_ds), len(dataloaders.valid_ds), len(test_dl.dataset)
```

[ ]:
```
(344, 87, 87)
```

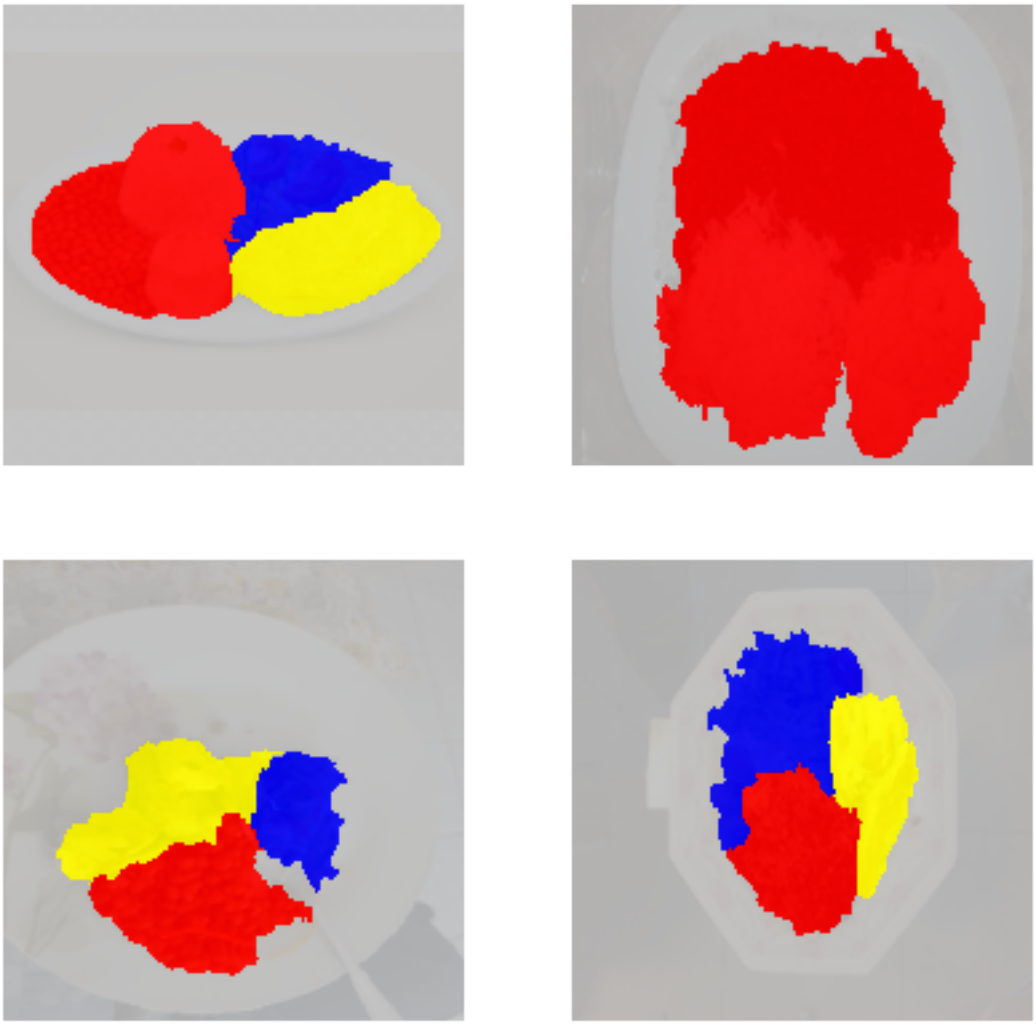
[ ]:
```
dataloaders.train.show_batch(max_n=4, figsize=(7, 7), cmap=cmap,␣
 ↪norm=norm,alpha=0.9)
```



[ ]:
```
dataloaders.valid.show_batch(max_n=4, figsize=(7, 7), cmap=cmap,␣
↪norm=norm,alpha=0.9)
```



[ ]:
```
test_dl.show_batch(max_n=4, figsize=(7, 7), cmap=cmap, norm=norm,alpha=0.9)
```



I.4.4 Metrics

[ ]:
```
from fastaiMetrics import acc_segmentation, MIOU

classes_index = range(1, len(codes)) #exclude background class at index 0
metrics = [acc_segmentation, DiceMulti, MIOU(classes_index, axis=1)]
```

I.4.5 Model

[ ]:
```
# EarlyStoppingCallback(monitor='miou',patience=50) -> if patience is too␣
 ↪high, this may cause error in learner.validate()
modelCallbacks = [ShowGraphCallback]
opt_func = Adam
```

```
# loss_func = CrossEntropyLossFlat(weight=lossClassWeights, axis=1)
loss_func = FocalLossFlat(weight=lossClassWeights, axis=1)

learner = unet_learner(dataloaders, resnet34, loss_func=loss_func,␣
↪opt_func=opt_func, metrics=metrics, cbs=modelCallbacks,
                       self_attention=False, act_cls=Mish).to_fp32()
```

[ ]:
```
#learner.model[0].load_state_dict(torch.load(path_to_drive + 'DiabetesLearning/
↪trainingConfig/resnet34_encoder-food101.h5'), strict=True); #load␣
↪preTrained from FOOD101
```

[ ]:
```
# learner.load(path + 'stage-2-brazilFoods', strict=False)
# learner.load_state_dict(torch.load(path + 'stage-2-brazilFoods.pth'),␣
↪strict=False)
# learner.model.load_state_dict(torch.load(path + 'stage-2-brazilFoods.pth'),␣
↪strict=False)
# test = Learner(dataloaders,resnet34)
learner.load('stage-1-best')
```

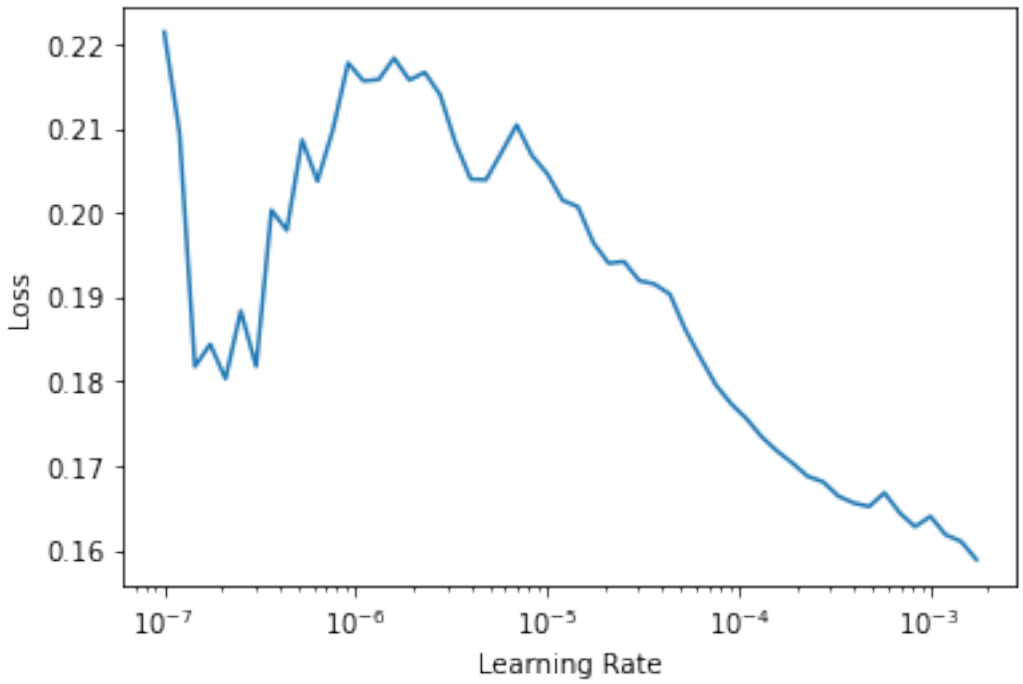
[ ]:
```
<fastai.learner.Learner at 0x7f78426d4e90>
```

I.4.6 Training

[ ]:
```
learner.lr_find()
```

```
<IPython.core.display.HTML object>
```

[ ]:
```
SuggestedLRs(lr_min=0.0001737800776027143, lr_steep=2.75422871709452e-06)
```



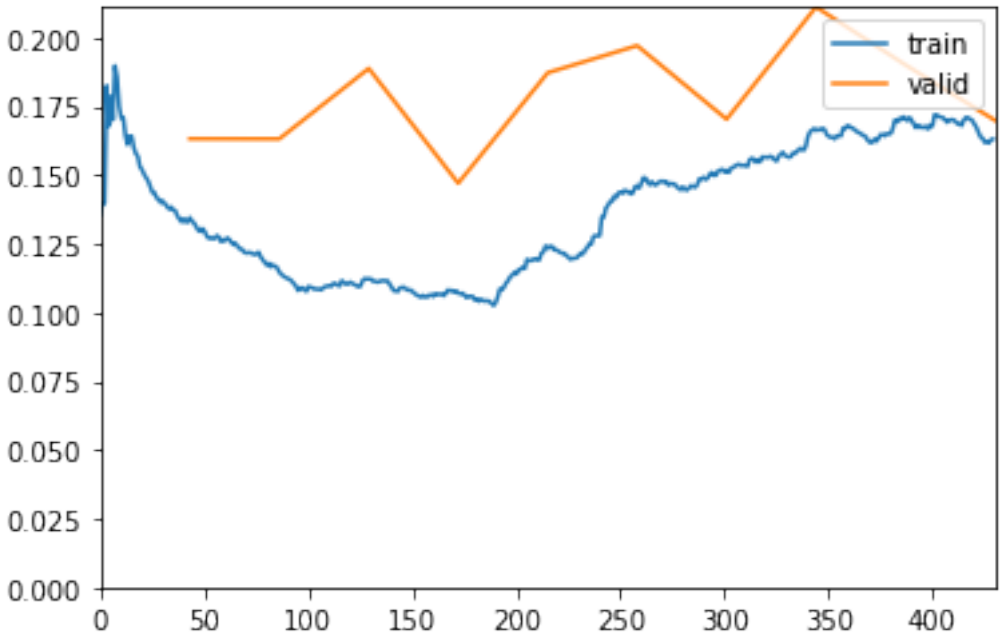
[ ]:
```
lr = 9e-4
freezeEpochs = 10
unfreezeEpochs = 10
wd = 1e-2
learner.freeze()
```

[ ]:
```
run = wandb.init(project="diabetesLearning") # track machine learning␣
 ↪experiment
fitCallbacks = [WandbCallback(log='all'), SaveModelCallback(every_epoch=False,␣
 ↪monitor='miou', fname='stage-1-best',with_opt=True),␣
 ↪GradientAccumulation(n_acc=gradientAcc)]
# learner.fit_one_cycle(epochs, slice(lr), pct_start=0.8, wd=wd,␣
 ↪cbs=fitCallbacks)
# learner.fit_flat_cos(epochs, slice(lr), wd=wd,cbs=fitCallbacks)
learner.fine_tune(unfreezeEpochs, base_lr=lr, freeze_epochs=freezeEpochs,␣
 ↪pct_start=0.3, wd=wd, cbs=fitCallbacks)
run.finish()
```

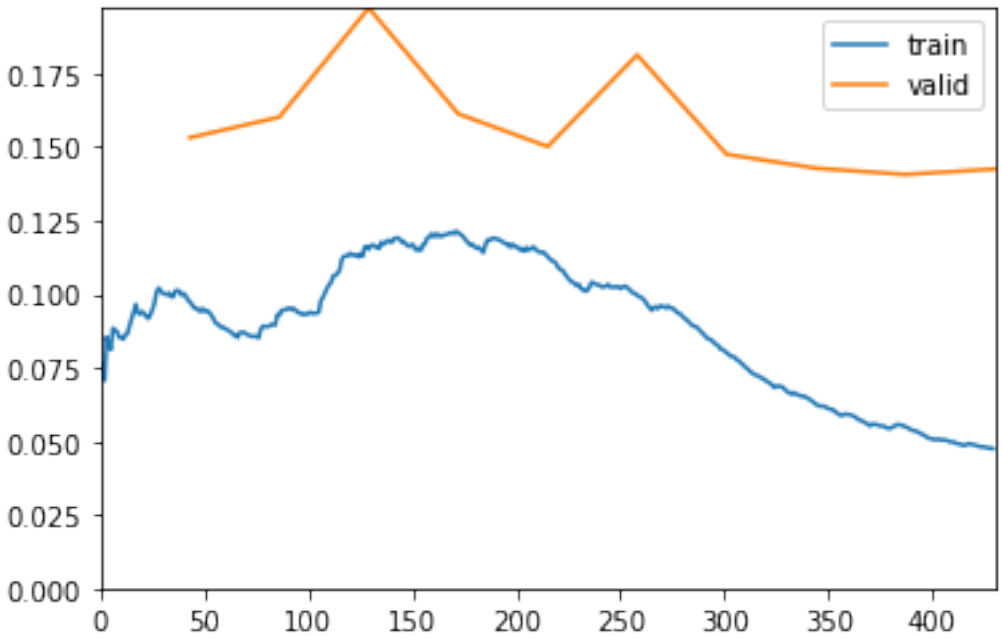
```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Better model found at epoch 0 with miou value: 0.5731584956649569.
```




```
Better model found at epoch 1 with miou value: 0.5755332698295655.
Better model found at epoch 3 with miou value: 0.6010744271245897.
<IPython.core.display.HTML object>
Better model found at epoch 0 with miou value: 0.6059478123610755.
```



```
Better model found at epoch 6 with miou value: 0.6204306771670653.
Better model found at epoch 7 with miou value: 0.6322319609794893.
```

```
<IPython.core.display.HTML object>
```

```
VBox(children=(Label(value=' 1008.91MB of 1009.17MB uploaded (0.00MB␣
 ↪deduped)\r'), FloatProgress(value=0.99974...
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

I.4.7 Fine-tunning (optional)

[ ]:
```
# import gc
# torch.cuda.empty_cache()
# gc.collect()
```

[ ]:
```
# learner = unet_learner(dataloaders, resnet34,␣
 ↪loss_func=CrossEntropyLossFlat(axis=1), metrics=metrics, cbs=callbacks,
#                        wd_bn_bias=True).to_fp16()
# learner.load('stage-1-best')
```

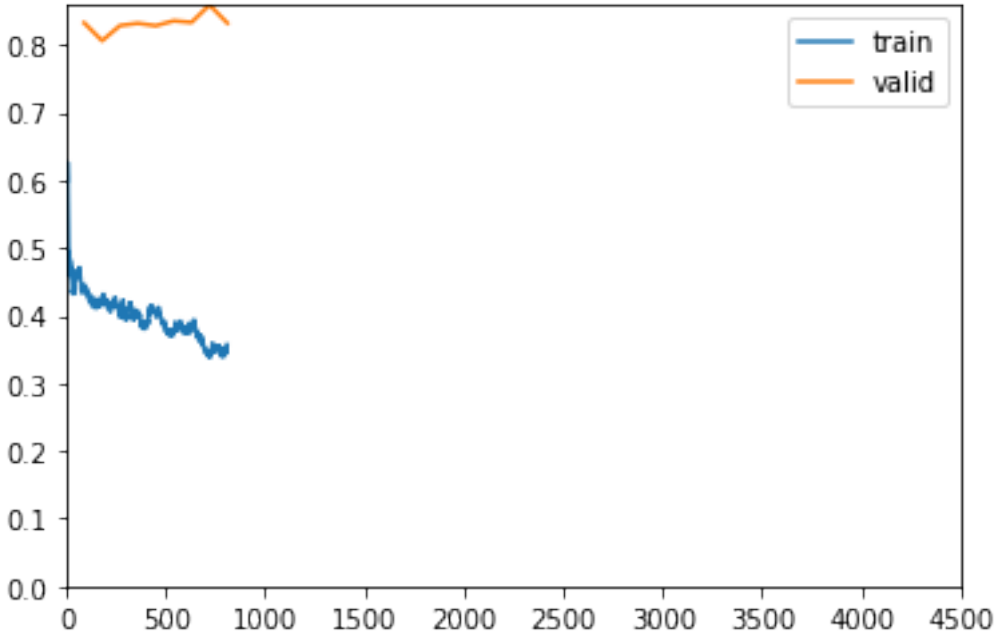
[ ]:
```
learner.unfreeze()
# learner.lr_find()
```

[ ]:
```
lrs = slice(lr/400,lr/4)
epochs = 50
wd = 1e-2
```

[ ]:
```
#run = wandb.init(project="diabetesLearning") # track machine learning␣
 ↪experiment
fitCallbacks = [WandbCallback(log='all'), SaveModelCallback(every_epoch=False,␣
 ↪monitor='acc_segmentation',␣
 ↪fname='stage-2-best',with_opt=True),GradientAccumulation(n_acc=gradientAcc)]
# learner.fit_one_cycle(epochs, lrs, pct_start=0.3, wd=wd, cbs=fitCallbacks)
learner.fit_flat_cos(epochs, lrs, wd=wd, cbs=fitCallbacks)
run.finish()
```

```
<IPython.core.display.HTML object>
```

```
/content/drive/My Drive/Colab Notebooks/DiabetesLearning/fastaiMetrics.py:39:
RuntimeWarning: invalid value encountered in true_divide
  iou_index = per_class_TP / (per_class_TP + per_class_FP + per_class_FN)

Better model found at epoch 0 with acc_segmentation value: 0.5323270559310913.
```




```
Better model found at epoch 1 with acc_segmentation value: 0.5447819232940674.
No improvement since epoch 1: early stopping

<IPython.core.display.HTML object>

VBox(children=(Label(value=' 1645.50MB of 1645.50MB uploaded (0.00MB␣
 ↪deduped)\r'), FloatProgress(value=1.0, ma...

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

[ ]:
```
# learner.load('stage-2-best')
# learner.save('stage-2-size350-bestEpoch')
```

I.4.8 FastAI Validation

[ ]:
```
id2name = {k:v for k,v in enumerate(codes)} # faz uma lista que relaciona nome
 ↪e id

#### fastai metrics names
metricsNames = ["Loss", "Acc_Segmentation", "Dice", "Miou"]

#### add iou per class if you want
from fastaiMetrics import IOU
iouPerClass = []
for x in range(1,len(codes)): iouPerClass.append(IOU(x, codes[x], axis=1,
 ↪ignore_index=0)) #ignore background idx 0
learner.metrics = metrics + iouPerClass
metricsNames += ["Iou " + codes[x] for x in range(1,len(codes))]

#### LOAD ONLY IF NECESSARY
# learner = unet_learner(dataloaders, resnet34, metrics=metrics+iouPerClass)
# learner.load('stage-1-best')
```

[ ]:
```
#### check if model overfits ####
trainset = tuple(zip(metricsNames,learner.validate(dl=dataloaders.train)))

#### validate validset ####
validset = tuple(zip(metricsNames,learner.validate(dl=dataloaders.valid)))

"Trainset", trainset, "Validset", validset
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

[ ]:
```
('Trainset',
 (('Loss', 0.02755793370306492),
  ('Acc_Segmentation', 0.9537791013717651),
  ('Dice', 0.9573008065624846),
  ('Miou', 0.9422343328130169),
  ('Iou Carbs', 0.9327461677416321),
  ('Iou Proteina', 0.9141959532173162),
  ('Iou Vegetais', 0.9314359032125074),
  ('Iou Frutas', 0.921803159355188)),
 'Validset',
 (('Loss', 0.15403492748737335),
```

```
('Acc_Segmentation', 0.8400511741638184),
('Dice', 0.7950899679457022),
('Miou', 0.6651723648813872),
('Iou Carbs', 0.7581479285091804),
('Iou Proteina', 0.6981870425535122),
('Iou Vegetais', 0.7938424216521796),
('Iou Frutas', 0.3190084457606669)))
```

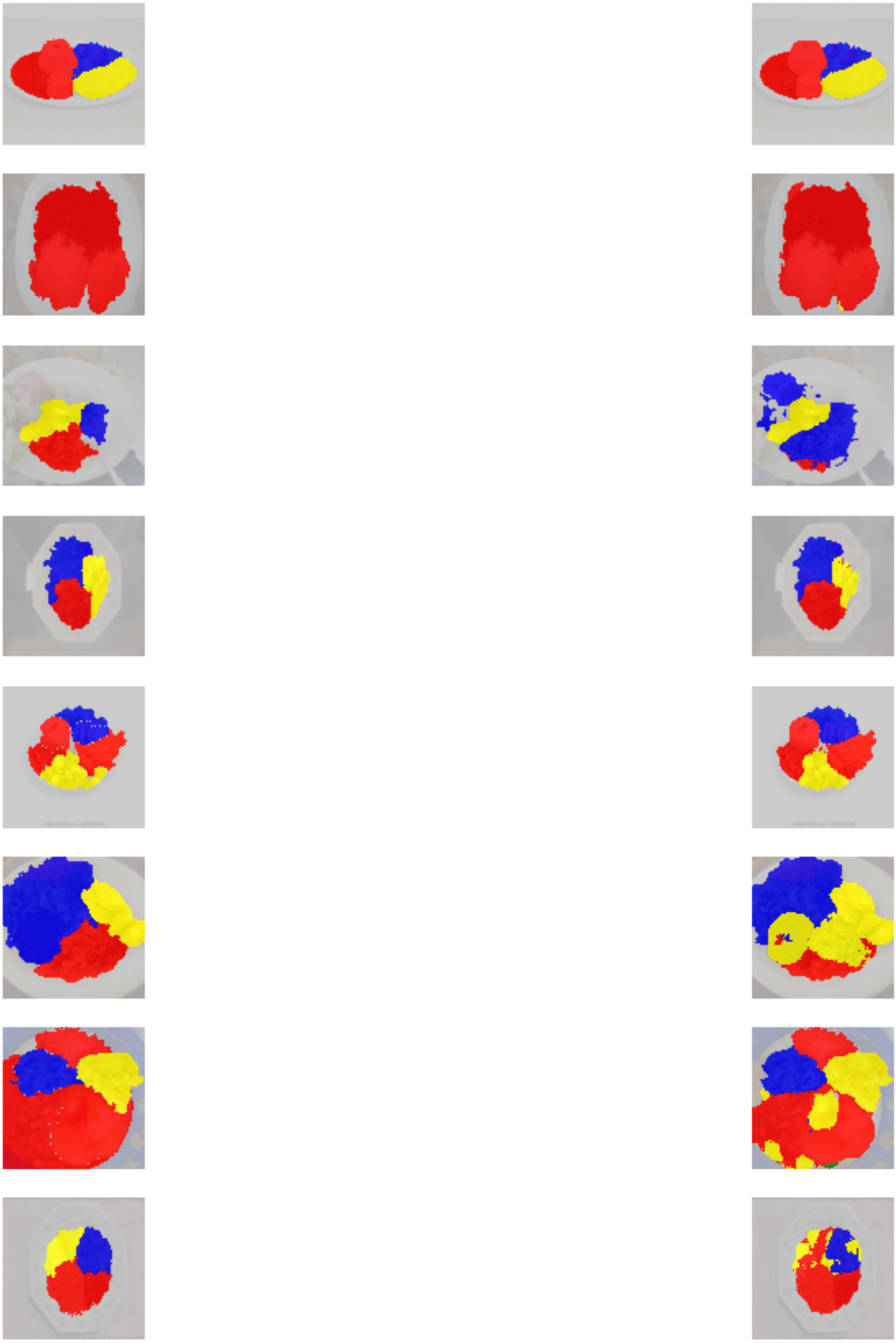
[ ]:
```
#### validate testset ####
# learner.dls.valid = test_dl
# testset = tuple(zip(metricsNames,learner.validate(dl=test_dl)))
# "Testset", testset
```

[ ]:
```
dl = dataloaders.valid
```

[ ]:
```
learner.show_results(dl=dl, figsize=(15,15), max_n=40, cmap=cmap,␣
↪norm=norm,alpha=0.8)
```

```
<IPython.core.display.HTML object>
```

Target/Prediction



[ ]:
```
# interp = Interpretation.from_learner(learner, dl=test_dl)
# losses,idxs = interp.top_losses()
# interp.plot_top_losses(4, figsize=(15,11), cmap=cmap, norm=norm)
```

[ ]:
```
#interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
#interp.most_confused(min_val=10)
```

I.4.9 Sklearn Validation

[ ]:
```
from sklearnMetrics import Metrics
void_code = 0
metricsObject = Metrics(codes, [void_code])
id2name = {k:v for k,v in enumerate(codes)} # faz uma lista que relaciona nome␣
 ↪e id

# LOAD ONLY IF NECESSARY
# learner = unet_learner(dataloaders, resnet34, metrics=metrics)
# learner.load('stage-1-best')
```

[ ]:
```
#input, probabilities, groundTruth, decoded, losses = learner.get_preds(dl=dl,␣
 ↪ with_input=True, with_loss=True, with_decoded=True, act=F.softmax)␣
 ↪#inputs, preds, groundTruth, losses
input, probabilities, groundTruth, decoded = learner.get_preds(dl=dl, ␣
 ↪with_input=True, with_loss=False, with_decoded=True, act=F.softmax)␣
 ↪#inputs, preds, groundTruth, losses
```

```
<IPython.core.display.HTML object>
```

```
/usr/local/lib/python3.7/dist-packages/fastai/learner.py:255: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
  res[pred_i] = act(res[pred_i])
```

[ ]:
```
print(metricsObject.getAccuracy(decoded.flatten().numpy(), groundTruth.
 ↪flatten().numpy()))
```

```
0.8394657058144419
```

[ ]:
```
print(metricsObject.get_f1Score(decoded.flatten().numpy(), groundTruth.
 ↪flatten().numpy()))
```

```
0.8410215281288161
```

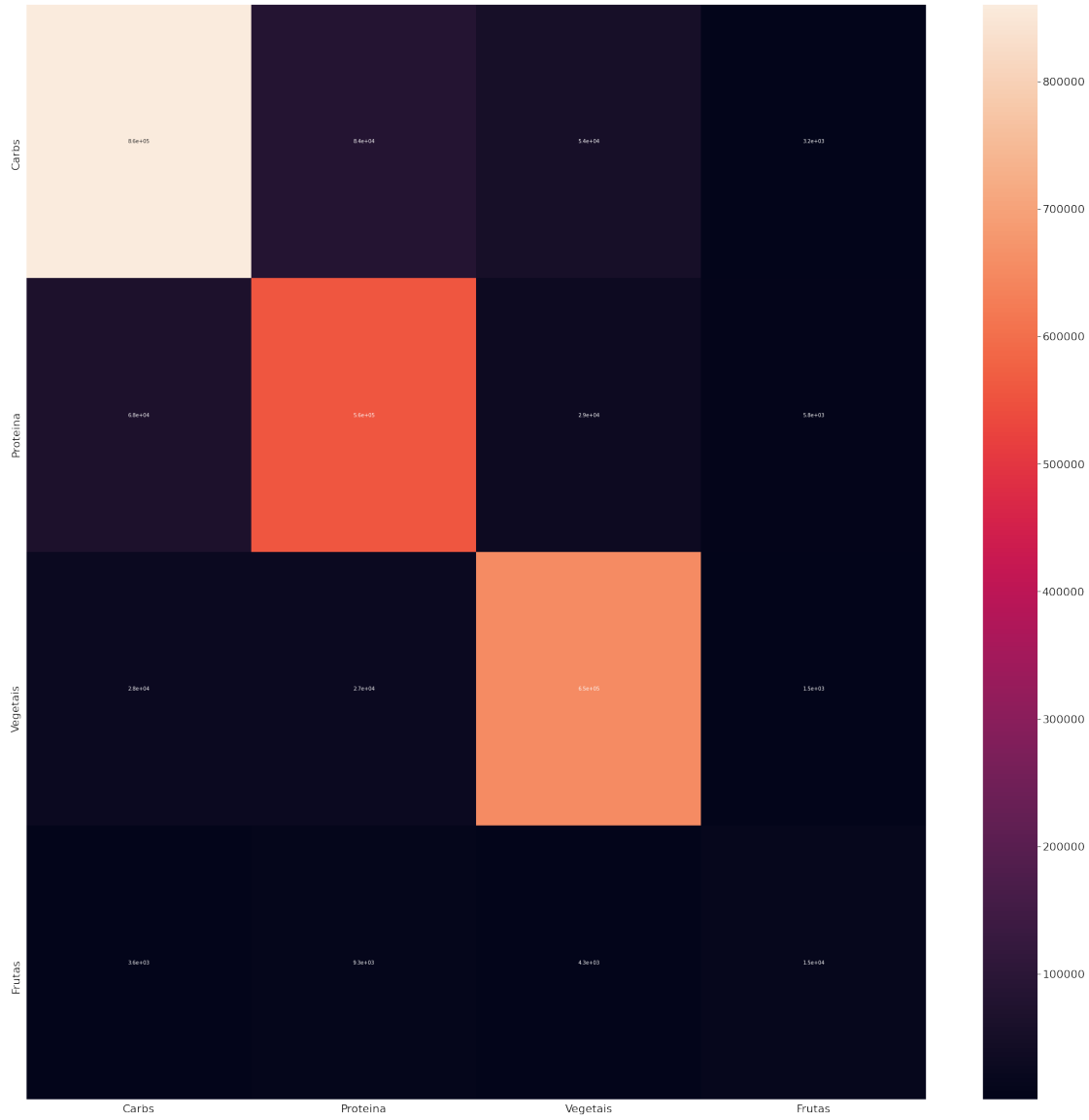
[ ]:
```
confusionMatrix = metricsObject.getConfusionMatrix(decoded.flatten().numpy(),␣
↪groundTruth.flatten().numpy(), plot=False)
```

[ ]:
```
mostConfused = metricsObject.mostConfused(confusionMatrix,␣
↪numberOfConfusions=4)

print("Actual", "   Predicted", "   Wrong pixels\n")
mostConfused
```

```
Actual    Predicted    Wrong pixels
```

[ ]:
```
[['Carbs', 'Proteina', 83688],
 ['Proteina', 'Carbs', 68403],
 ['Carbs', 'Vegetais', 54325],
 ['Proteina', 'Vegetais', 28587]]
```

[ ]:
```
#plot confusion matrix (take some time, too big the matrix). Think in a better␣
↪way to visualize (check for most confuseds)

metricsObject.plotConfusionMatrix(confusionMatrix, font_scale=2,␣
↪removeDiagonal=False)
```



[ ]:
```
print(metricsObject.getClassificationReport(decoded.flatten().numpy(),
↪groundTruth.flatten().numpy()))
```

```
              precision    recall  f1-score   support

       Carbs       0.87      0.83      0.85   1035158
    Proteina       0.81      0.82      0.82    676648
   Vegetais        0.85      0.89      0.87    736392
     Frutas        0.53      0.41      0.47     35153

   micro avg       0.84      0.84      0.84   2483351
   macro avg       0.77      0.74      0.75   2483351
weighted avg       0.84      0.84      0.84   2483351
```

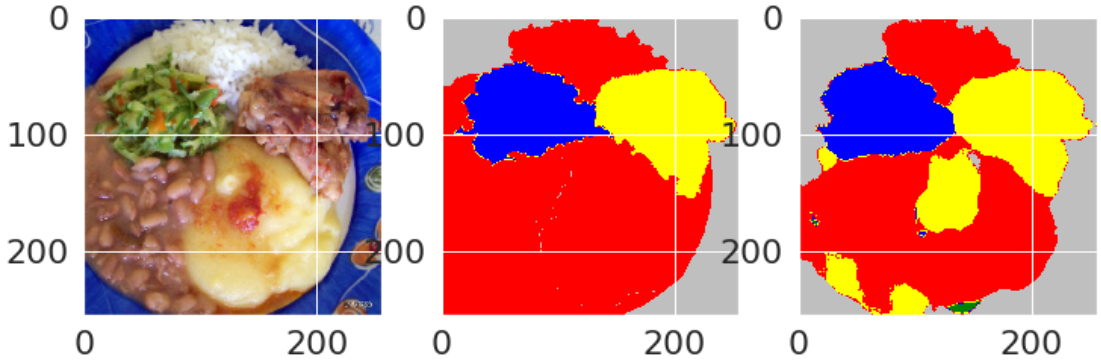
I.4.10 Check some particular image details

[ ]:
```
from collections import OrderedDict

id2name = {k:v for k,v in enumerate(codes)} # faz uma lista que relaciona nome␣
↪e id

def getFoodsInImage(tensor, removeVoid=False):
    foods, counts = np.unique(tensor,return_counts=True)
    if removeVoid:
        counts = np.delete(counts, 0) #tira o void na posicao 0 do array
        foods = np.delete(foods, 0) #tira o void na posicao 0 do array

    foodNames = []
    print(f'Numero de alimentos: {len(foods)}')
    for food in foods:
        foodNames = np.append(foodNames,id2name[food])
    dictCounts = dict(zip(foodNames, counts))
    dictCounts = OrderedDict(sorted(dictCounts.items(), key=lambda x: x[1],␣
↪reverse=True))
    print(dictCounts)
    return dictCounts

def getMostAppearances(foodCountsDict, k = 3):
    sortedDict = OrderedDict(sorted(foodCountsDict.items(), key=lambda x:␣
↪x[1], reverse=True))
    topKDict = {}
    for count, (key, v) in enumerate(foodCountsDict.items()):
        if count == k:
            break
        elif key != "NaoAlimento":
            topKDict[key] = v
        else:
            k += 1 # increment k if find NaoAlimento in the first positions

    return topKDict
```

[ ]:
```
idx = 6
fig = plt.figure(figsize=(12,12))
```

```
# denorm to original image
rawImage = copy(input[idx])

nrm = Normalize.from_stats(*imagenet_stats)
image = nrm.decode(rawImage) #fastai version 2.3.1 needs two denorms!!!! FUCK␣
 ↪YOU FASTAI!!!
image = nrm.decode(image).clamp(0,1).squeeze().permute(1,2,0) # denorm image␣
 ↪from imagenet stats

ax1 = fig.add_subplot(1,3,1)
ax1.imshow(image)

ax2 = fig.add_subplot(1,3,2)
ax2.imshow(groundTruth[idx], cmap=cmap, norm=norm)

ax3 = fig.add_subplot(1,3,3)
ax3.imshow(decoded[idx], cmap=cmap, norm=norm)
```

[ ]: <matplotlib.image.AxesImage at 0x7fb08d69ab10>



[ ]:
```
print("Ground truth\n")
sample = groundTruth[idx]
gtDictCounts = getFoodsInImage(sample)
```

```
Ground truth

Numero de alimentos: 4
OrderedDict([('Carbs', 36171), ('NaoAlimento', 14300), ('Proteina', 8115),
('Vegetais', 6950)])
```

[ ]:
```
print("Prediction\n")
sample = decoded[idx]
decodedDictCounts = getFoodsInImage(sample)
```

```
Prediction

Numero de alimentos: 5
OrderedDict([('Carbs', 25732), ('NaoAlimento', 19203), ('Proteina', 13139),
('Vegetais', 7257), ('Frutas', 205)])
```

[ ]:
```
# get low prob classes

bestProbs, bestProbsIdx = torch.max(probabilities[idx], dim=0)
lowProbs = []
lowProbsIdx = []

mask = bestProbs < 0.5
lowProbsIdx = bestProbsIdx[mask]
lowProbDictCounts = getFoodsInImage(lowProbsIdx)

print("\nPercentual de pixeis com baixa probabilidade\n")
for key, value in lowProbDictCounts.items():
    pct = float(value/decodedDictCounts[key]*100)
    print(key, '{:.2f}'.format(pct), "%" )
```

```
Numero de alimentos: 5
OrderedDict([('Carbs', 15695), ('Proteina', 6658), ('NaoAlimento', 3616),
('Vegetais', 2058), ('Frutas', 205)])

Percentual de pixeis com baixa probabilidade

Carbs 60.99 %
Proteina 50.67 %
NaoAlimento 18.83 %
Vegetais 28.36 %
Frutas 100.00 %
```

[ ]:
```
# get high prob classes

bestProbs, bestProbsIdx = torch.max(probabilities[idx], dim=0)

mask = bestProbs > 0.5
```

```
highProbsIdx = bestProbsIdx[mask]
highProbDictCounts = getFoodsInImage(highProbsIdx)

print("\nPercentual de pixeis com alta probabilidade\n")
for key, value in highProbDictCounts.items():
    pct = float(value/decodedDictCounts[key]*100)
    print(key, '{:.2f}'.format(pct), "%" )
```

```
Numero de alimentos: 5
OrderedDict([('Carbs', 24024), ('NaoAlimento', 18602), ('Proteina', 11822),
('Vegetais', 6995), ('Frutas', 80)])

Percentual de pixeis com alta probabilidade

Carbs 93.36 %
NaoAlimento 96.87 %
Proteina 89.98 %
Vegetais 96.39 %
Frutas 39.02 %
```

I.4.11 Predictions of new input data

[ ]:
```
# always load it!

learner = unet_learner(dataloaders, resnet34, self_attention=False,␣
 ↪act_cls=Mish)
learner.load('stage-1-best')
```

[ ]:
```
<fastai.learner.Learner at 0x7f833e0a65d0>
```

[ ]:
```
#input should be a PIL object

index = 50
img = dataloaders.valid_ds[index][0]

# img_pil = Image.open("3.jpg")
# img = np.asarray(img_pil.resize((400,400)))

pred = learner.predict(img)

fig = plt.figure(figsize=(12,12))
ax1 = fig.add_subplot(1,2,1)
```

```
ax1.imshow(pred[0],cmap=cmap)

ax2 = fig.add_subplot(1,2,2)
ax2.imshow(img)

foods = getFoodsInImage(pred[0])
k = 3
mostAppear = list(getMostAppearances(foods, k=k).keys())
print('\n{} principais alimentos:'.format(k),mostAppear)
```

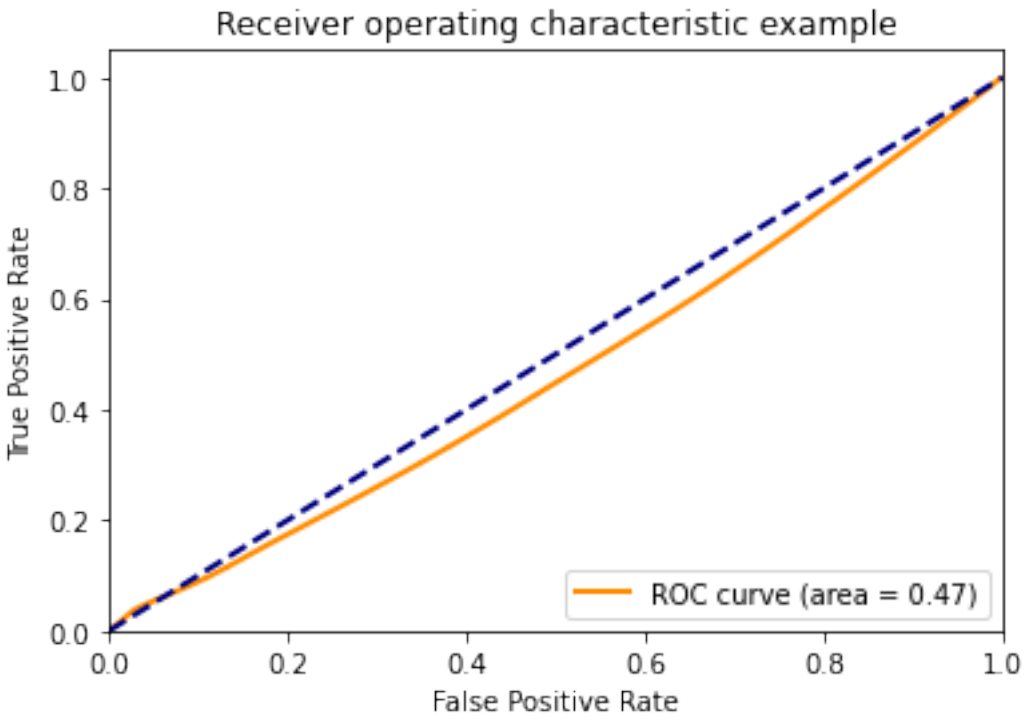
[ ]:
```
from sklearn.preprocessing import label_binarize

mask = groundTruth.unsqueeze(dim=1)!=0
gtOneHotEncoded = label_binarize(groundTruth.flatten().numpy(),␣
 ↪classes=range(len(codes)))

# groundTruthIgnoringBackground = groundTruth[mask.squeeze()]
# gtOneHotEncoded = label_binarize(groundTruthIgnoringBackground.flatten().
 ↪numpy(), classes=range(len(codes)))

# probabilitiesIgnoringBackground = probabilities[mask.unsqueeze(dim=1)]
probabilities = probabilities.reshape((-1,16,256*256))
probabilities = probabilities.permute(0,2,1)
probabilities = probabilities.reshape((-1,16))

flatMask = mask.flatten()
# probabilitiesIgnoringBackground = np.array([Tensor(probabilities[i,:]) for i␣
 ↪in range(len(flatMask[:100000])) if flatMask[i] == True])
# probabilitiesIgnoringBackground = probabilities[mask.flatten().
 ↪unsqueeze(dim=1)]

flatMask.shape, probabilities.shape, gtOneHotEncoded.shape
```

[ ]:
```
(torch.Size([5636096]), torch.Size([5636096, 16]), (5636096, 16))
```

[ ]:
```
# Compute micro-average ROC curve and ROC area
fpr, tpr, _ = roc_curve(gtOneHotEncoded.ravel(), probabilities.ravel())
roc_auc = auc(fpr, tpr)
```

[ ]:
```
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
```

```
        lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



[ ]: