

UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Modelo de Deep Learning para Reconhecimento de Frutos
de Cacau

Juliana Rodrigueiro Clavisio Pereira de Oliveira



São Carlos – SP

Modelo de Deep Learning para Reconhecimento de Frutos de Cacau

Juliana Rodrigueiro Clavissio Pereira de Oliveira

Orientador: Profa. Dr. Roseli Ap. Francelin Romero

Monografia final de conclusão de curso apresentada
ao Instituto de Ciências Matemáticas e de
Computação – ICMC-USP, como requisito
parcial para obtenção do título de Engenheira de
Computação.

Área de Concentração: Inteligência Computacional

**USP – São Carlos
Novembro de 2017**

Este trabalho é dedicado aos trabalhadores invisíveis que fazem de seu suor o conforto alheio.

AGRADECIMENTOS

Agradeço à Professora Roseli, que prontamente e com muita generosidade aceitou orientar-me mesmo em condições não usuais e sempre que necessário direcionou-me ao melhor caminho para atingir bons resultados neste trabalho.

Tenho também muito a agradecer a Universidade de São Paulo por ter proporcionado os melhores anos de minha vida, e espero um dia poder retribuir, de alguma forma, todo o conhecimento recebido.

Impossível não mencionar pessoas excepcionais que auxiliaram-me nesta longa jornada, como o Professor Santana, que sempre comprehende seus alunos e procura ajudar todos da melhor forma possível. E também meus pais, Helio e Elsa, e meu irmão, André, que com muito trabalho, paciência, amor e humildade forneceram-me as possibilidades que tenho hoje.

Aos meus amigos da Engenharia de Computação que sabem na pele os prazeres e pesares vivenciados durante os anos de graduação.

Finalmente, agradeço ao meu parceiro e amigo Scott James, pois sem suas ideias de como aperfeiçoar o banco de dados, talvez este trabalho não teria obtido resultados tão bons.

“We are but dwarfs standing upon the shoulders of giants. We see more, and things that are more distant, than they, but not because our sight is superior; or because we are taller than they, but because they raise us up, and by their great stature add to ours.”

(John of Salisbury)

RESUMO

OLIVEIRA, J. R. C. P.. **Modelo de Deep Learning para Reconhecimento de Frutos de Cacau**. 2017. 56 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

O aprendizado de máquina tornou-se o catalisador predominante de automação nos últimos anos. Muitos modelos sofisticados de redes convolucionais profundas tem seu código aberto e é acessível a todos, o que faz com que, através de transferência de aprendizado, abram-se oportunidades para criar projetos de automação focados e específicos para cada indústria. Devido a diversos obstáculos, a automação convencional de certas indústrias agrícolas de grande escala foi deixada fora de questão até mesmo para mecanizações simples. Isso leva a quantidades de rendimento fracas, bem como um impacto sobre os padrões nos quais os trabalhadores manuais vivenciam dentro dessas indústrias. Esse trabalho apresenta o primeiro passo do processo de automatização da colheita dos frutos de cacau em suas plantações, dentro da proposta de eventualmente existirem colheitadeiras robóticas que possam coletar e transportar produtos. O foco está no passo inicial de treinar um modelo de aprendizagem convolucional profunda para identificar os frutos de cacau, usando técnicas de transferência de aprendizagem de última geração. Este trabalho será eventualmente expandido e utilizado para circumavegar os artifícios que, até agora, permitiram a indústria do cacau trabalhar apenas em padrões pré-industriais.

Palavras-chave: *Deep learning, Transferência de aprendizado, Redes neurais convolucionais, Classificação de imagens, Cacau.*

ABSTRACT

OLIVEIRA, J. R. C. P.. **Modelo de Deep Learning para Reconhecimento de Frutos de Cacau**. 2017. 56 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Machine learning has become a prevalent catalyst of automation in recent years. Many sophisticated convolutional deep learning models are open source and accessible to all, which, using transfer learning, opens an opportunity to create focused, industry-specific automation projects. Due to many obstructions, the conventional automation of certain large scale agricultural industries by simple mechanisation has been left out of question. This leads to poor yield quantities, as well as an impact upon the standards at which manual workers experience within said industries. This paper introduces the first step of the process in automating the harvesting of cocoa pods on cocoa plantations, within a proposal to eventually have automated robotic workers which can gather and transport produce. The focus is on the initial step of training a convolutional deep learning model to identify cocoa bean pods, using the state-of-the-art transfer learning techniques. It will be eventually built upon and used to circumnavigate the pitfalls which has, until now, left the cocoa industry working at pre-industrial standards.

Key-words: Deep learning, Transfer Learning, Convolutional neural networks, Image classification, Cocoa.

LISTA DE ILUSTRAÇÕES

Figura 1 – Comparação entre algoritmos de otimização.	28
Figura 2 – Módulo <i>Inception</i>	31
Figura 3 – Rede <i>Inception-Resnet-v2</i>	32
Figura 4 – Esquema usado para aplicar peso aos resíduos.	33
Figura 5 – Resultados Top-1 por rede.	34
Figura 6 – Comparação entre acurácia, operações e quantidade de parâmetros. O tamanho da circunferência é proporcional ao número de parâmetros da rede.	35
Figura 7 – Fonte das imagens que compõem o conjunto de dados.	39
Figura 8 – Quadros-chave de um vídeo sobre frutos de cacau para garantir um equilíbrio de variedade.	39
Figura 9 – Exemplos de imagens com cacau que compõem o banco de dados deste trabalho.	40
Figura 10 – Gráfico da acurácia por iterações.	42
Figura 11 – Gráfico da precisão por iterações.	42
Figura 12 – Gráfico da revocação por iterações.	43
Figura 13 – Gráfico do coeficiente MCC por iterações.	43
Figura 14 – Soma da quantidade de falsos negativos durante o treinamento.	44
Figura 15 – Soma da quantidade de falsos positivos durante o treinamento.	44
Figura 16 – Soma da quantidade de verdadeiros negativos durante o treinamento.	45
Figura 17 – Soma da quantidade de verdadeiros positivos durante o treinamento.	45
Figura 18 – Representação espacial das previsões durante o treino parcial. No eixo y encontram-se as quantidades de exemplos, no eixo x as probabilidades e no eixo z os passos do treinamento	46
Figura 19 – Representação espacial das previsões durante o treino total. No eixo y encontram-se as quantidades de exemplos, no eixo x as probabilidades e no eixo z os passos do treinamento	46
Figura 20 – Comparação do coeficiente MCC durante o treino total com e sem o ponto de inspeção.	47
Figura 21 – Ausência de cacau: 90,96%	55
Figura 22 – Ausência de cacau: 99,22%	55
Figura 23 – Presença de cacau: 97,91%	56
Figura 24 – Presença de cacau: 92,81%	56

LISTA DE TABELAS

Tabela 1 – Matriz de Confusão	29
Tabela 2 – Tempo de treinamento.	41
Tabela 3 – Comparação do coeficiente MCC durante treino e avaliação.	45

LISTA DE ABREVIATURAS E SIGLAS

- BP *Back-Propagation*
CAM *Content Addressed Memory*
CNN *Convolutional Neural Network*
CPU *Central Processing Unit*
GPU *Graphics Processing Unit*
ILSVRC .. *ImageNet Large Scale Visual Recognition Challenge*
MCC Coeficiente de Correlação Matthews
MLP *Multi-Layer Perceptrons*
ReLU *Rectified Linear Unit*
RNA Redes Neurais Artificiais

SUMÁRIO

1	INTRODUÇÃO	21
1.1	<i>Motivação e Contextualização</i>	21
1.2	<i>Objetivos</i>	22
1.3	<i>Organização</i>	22
2	REVISÃO BIBLIOGRÁFICA	23
2.1	<i>Considerações Iniciais</i>	23
2.2	<i>Aprendizado de Máquina</i>	23
2.2.1	<i>Tipos de Aprendizado</i>	23
2.2.2	<i>Redes Neurais Artificiais</i>	24
2.2.3	<i>Redes Profundas - Deep Learning</i>	25
2.2.4	<i>Redes Neurais Convolucionais</i>	25
2.3	<i>Transferência de Aprendizado</i>	26
2.4	Treinamento	27
2.4.1	<i>Classificação</i>	27
2.4.2	<i>Função de Custo</i>	27
2.4.3	<i>Otimização</i>	27
2.5	Avaliação	28
2.5.1	<i>Matriz de Confusão</i>	29
2.5.2	<i>Acurácia</i>	29
2.5.3	<i>Precisão</i>	29
2.5.4	<i>Revocação</i>	30
2.5.5	<i>Coeficiente de Correlação Matthews</i>	30
2.6	Modelos	30
2.6.1	<i>Inception</i>	30
2.6.2	<i>Redes Residuais</i>	31
2.6.3	<i>Inception-Resnet-v2</i>	32
2.6.3.1	<i>Arquitetura</i>	32
2.6.3.2	<i>Banco de Dados do Pré-Treino</i>	33
2.6.3.3	<i>Comparações</i>	33
2.7	Considerações Finais	34
3	DESENVOLVIMENTO	37

3.1	Considerações Iniciais	37
3.2	Descrição da Pesquisa Proposta	37
3.3	Materiais	37
3.4	Base de Dados	38
3.5	Descrição das Atividades Realizadas	40
3.6	Resultados	41
3.7	Dificuldades e Limitações	46
4	CONCLUSÃO	49
4.1	Contribuições	49
4.2	Relacionamento entre o Curso e o Projeto	49
4.3	Considerações sobre o Curso de Graduação	49
4.4	Trabalhos Futuros	50
REFERÊNCIAS		53
APÊNDICE A	EXEMPLOS E PROBABILIDADES	55

Capítulo 1

INTRODUÇÃO

1.1 Motivação e Contextualização

Atualmente a tecnologia atende à função mais integral na maioria das indústrias. Durante décadas, a automação aumentou a produtividade e a qualidade do trabalho para os trabalhadores dentro da indústria automotiva, agricultura, construção e inúmeros outros setores. A automação, como é amplamente entendido, é a substituição de tarefas comuns e repetitivas, geralmente em uma linha de produção dentro de fábricas ou em agricultura de grande escala.

Até poucos anos, o trabalhador humano permaneceu como um componente necessário de um processo automatizado. Excluindo-se os engenheiros que projetam e mantêm *hardware* e *software*, as pessoas qualificadas devem estar presentes para realizar tarefas que exigem um nível de cognição abstrata como garantia de qualidade. Este gargalo criado pela necessidade da presença humana, no entanto, começou a expandir-se recentemente. À medida que o aprendizado de máquina se torna cada vez mais bem-sucedido, as novas possibilidades de automação já começam a ser exibidas. Os limites estabelecidos pela programação estática convencional estão sendo ampliados pelos resultados impressionantes dos projetos de aprendizado de máquina. Nos últimos anos, vimos os maiores mestres de xadrez e *Go* do mundo derrotados por *softwares* de inteligências artificiais, várias organizações começaram a trabalhar em carros autodirigidos completamente automáticos, além da excelência em reconhecimento facial conseguida pelo *Facebook*, dentre outros.

Conforme referido anteriormente, a automação no mundo da agricultura progrediu em passos sem precedentes ao longo dos anos. No entanto, o elemento humano dentro deste campo pode ser vital por vários motivos. De um toque delicado, à intuição de saber exatamente quando a produção está pronta para ser colhida, os seres humanos ainda têm a vantagem em máquinas, por enquanto. Além disso, muitas formas de produção são cultivadas em regiões subdesenvolvidas, onde os custos de automação ainda são muito superiores em comparação aos custos de雇用 trabalhadores com baixa remuneração para atingir metas de produção viáveis.

As plantações de cacau fornecem um desafio único para a automação. As árvores de cacau não podem crescer em formas regulares, e o produto (o fruto do cacau) cresce diretamente do tronco da árvore, muitas vezes obstruído por galhos e folhagem. Além disso, o cultivo não pode ser feito em fileiras, com grandes espaços abertos, dificultando até a mecanização básica tanto para a colheita quanto para o transporte. Considerando todos esses fatores, juntamente com

a necessidade de mão de obra humana (cada vez mais escassa) para identificar frutos maduros, cortá-los e transportá-los, a colheita de cacau permanece até hoje uma tarefa que requer um trabalho manual extenuante.

Pelo que é do nosso conhecimento, ainda não existe um sistema proposto de realização de automação para plantações de cacau. Uma ideia inicial proposta pelo Laboratório de Aprendizado de Robôs do ICMC-USP , é fazer uso de um *drone* combinado com colheitadeiras de pequeno porte. O *drone* navegaria sobre a folhagem irregular da plantação, identificando frutos de cacau maduros e cortando-os da árvore, onde então, um humano ou possivelmente, em etapas posteriores, colheitadeiras menores, capazes de navegar pelo terreno acidentado dentro da plantação, coletaria os frutos. Usando algoritmos de aprendizado da máquina, boa parte deste desafio poderá ser solucionado, porém exigirá muitas inovações para torná-lo operacional com funcionalidade para plantações de cacau especificamente.

1.2 Objetivos

Este trabalho é o passo inicial para a realização do sistema proposto de automação de uma plantação de cacau. O requisito mais importante do *drone* de colheita é a sua capacidade de identificar de forma precisa e confiável um fruto de cacau dentro de seu ambiente. Utilizando técnicas de aprendizado de máquina, tais como redes neurais profundas (*Deep Learning*) para o reconhecimento de imagem, um modelo sofisticado foi treinado para a identificação de cacau. Nesta fase inicial, o objetivo é concentrar-se em garantir que o modelo possa identificar com precisão a presença de um fruto a partir de uma imagem estática, o mais próximo possível de 100%.

Como a habilidade de separar um fruto de cacau de todos os outros objetos em vista é uma característica fundamentalmente essencial para que todo o sistema funcione, o foco deste trabalho é, sobretudo, uma resposta binária confiável. É sobre esta base que será construída a capacidade de separar maduro de não maduro, de identificar o fruto em tempo real, e de localizar a haste para o corte.

1.3 Organização

No Capítulo 2 são descritas as bases teóricas em que este trabalho foi fundamentado, dando destaque para as técnicas aqui aplicadas. No Capítulo 3, serão explicados como foi feito o processo de aquisição dos dados e como foi implementada a transferência de aprendizagem das redes profundas. Por fim, no Capítulo 4 serão apresentadas as conclusões, os trabalhos futuros e considerações finais.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

2.1 Considerações Iniciais

Neste capítulo são introduzidos os conceitos de aprendizado de máquina e seus subtópicos relacionados a este trabalho, além da apresentação do tema Transferência de Aprendizado e como pode ser utilizado. São também discutidas as métricas utilizadas para treinamento e avaliação de desempenho de redes profundas. Por fim, são apresentados os modelos de *Deep Learning* utilizados no presente trabalho.

2.2 Aprendizado de Máquina

“Aprendizado de máquina” é um termo usado para descrever um sistema no qual um programa pode ter a habilidade de “aprender”. Dentro da programação clássica, um programa é limitado pelo que o engenheiro pode especificar explicitamente, usando instruções estáticas do programa, que são funcionais para tarefas que requerem resultados menos complexos, ou usando dados previsíveis.

Por outro lado, algoritmos de aprendizagem de máquinas são programas que agem como modelos que podem avaliar os dados, de forma automática, utilizando análise estatística e reconhecimento de padrões, que se presta a análises preditivas, ajudando muitos pesquisadores, cientistas de dados, engenheiros e analistas a alcançar padrões muito mais elevados por meio da automação, bem como descobrindo padrões nunca antes vistos dentro dos dados. O Aprendizado de Máquina é um termo abrangente e existem vários sistemas que se enquadram nesta categoria, sistemas que envolvem diferentes algoritmos para alcançar seus objetivos da maneira mais eficiente possível.

2.2.1 Tipos de Aprendizado

A maneira pela qual um computador é “ensinado” pode variar dentro do domínio da aprendizagem em máquina, dependendo da saída desejada, ou da natureza dos dados que se tem em mãos. O paradigma utilizado neste trabalho é referido como “aprendizado supervisionado”.

O aprendizado supervisionado é um sistema no qual apresenta-se ao modelo de aprendizagem de máquina, exemplos de dados predefinidos, isto é, pares de entrada e saída, denominados

dados rotulados, cujos rótulos diferem de uma classe para outra. Então, por meio da figura de um professor, o sistema de aprendizagem será treinado para aprender a reconhecer padrões ensinados, criando suas próprias regras e estabelecendo uma distinção entre as classes (HAYKIN, 1994). Este formato é usado para ensinar um computador a identificar e/ou categorizar dados que não podem ser definidos com restrições simples, por razões como uma grande e nebulosa quantidade de variação ou, no caso deste trabalho, identificar o conteúdo das imagens.

Na aprendizagem não supervisionada, por outro lado, os dados a serem apresentados ao modelo de aprendizagem não são rotulados. O algoritmo de aprendizado deve aprender a extrair características dos dados e estabelecer correlações entre eles, se existir alguma artificial. Depois disso, é necessário um outro processo para extração dos padrões aprendidos pelo modelo.

Alternativamente, há o aprendizado de reforço (*reinforcement learning*), o que é provavelmente o mais parecido com a forma como alguém treinaria uma inteligência orgânica. Este paradigma de aprendizado é muito utilizado na robótica, ou em qualquer forma de máquina que seja necessária a interação com ambientes dinâmicos. O processo de aprendizado ocorre por meio de tentativas de ações corretas ou incorretas, tomadas pelo robô, para as quais, reforços positivos ou negativo são recebidos pelo modelo de aprendizagem, respectivamente até que ele consiga interagir com o ambiente de forma satisfatória, isto é, quando o máximo dos reforços é conseguido.

2.2.2 Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) são modelos computacionais inspirados no funcionamento do nosso sistema nervoso, elas tentam emular as redes neurais biológicas. Uma rede neural consiste em neurônios artificiais, interligados (como em sinapses), permitindo que se comuniquem e passem informações entre si. Cada sinapse carrega um peso, que é continuamente modificado à medida que o sistema aprende as características presentes no conjunto de dados. As RNA são uma abordagem à computação que confere às máquinas a capacidade de processar e reconhecer dados de forma muito mais orgânica.

Existem diferentes modelos de RNA, dependendo da sua funcionalidade, topologia e algoritmo de aprendizado. As mais conhecidas são as redes neurais multi-camadas (em inglês: *Multi-Layer Perceptrons* (MLP)), com o algoritmo *Back-Propagation* (BP) (RUMELHART; HINTON; WILLIAMS, 1986), que podem ser utilizadas na maioria dos problemas de classificação de padrões. No entanto, elas também podem ser usadas como *Auto-encoders* e aproximação de funções. Existem redes com topologia de uma única camada, como por exemplo, as Redes de Hopfield, que foram propostas para resolver o problema do caixeiro viajante e o problema *Content Addressed Memory* (CAM) (HAYKIN, 1994). Ao processar dados suficientes (o que possivelmente equivale à grandíssimas quantidades), as redes neurais podem identificar padrões extremamente abstratos, o que seria de outrora impossível usando instruções de algoritmos estáticos convencionais.

Um dos principais problemas relacionado a RNA é o *overfitting*. Quando o modelo preditor se adapta perfeitamente nos exemplos de treinamento, porém obtém desempenho ruim no conjunto de teste. Isso geralmente acontece quando o modelo é muito complexo e se ajusta trivialmente aos dados, ou quando não há dados suficientes para estimar os parâmetros com precisão.

2.2.3 Redes Profundas - Deep Learning

“Rede neural profunda” é a terminologia utilizada para descrever uma rede neural com muitas camadas ocultas que separam a entrada e saída, criando profundidade. Quando os neurônios em cada uma das camadas de uma rede neural são ativados, melhor simula de que forma um neurônio biológico reage aos estímulos visuais. Os pesos das conexões entre todos os neurônios guiam a rede profunda a estabelecer gradualmente os recursos mais abstratos conforme os dados visuais são filtrados através das camadas. A partir da entrada, os neurônios identificam formas e ângulos e, à medida que os dados se movem para camadas mais profundas, o objeto aos poucos se torna mais reconhecível para a perspectiva humana. Eventualmente, nas camadas superiores, mais próximas da saída, os neurônios terão estabelecido todo o conteúdo lidando com os recursos mais abstratos como um todo.

Camadas *pooling* são frequentemente utilizadas em redes profundas com o objetivo de progressivamente reduzir a quantidade de recursos e a complexidade computacional da rede. O algoritmo mais comum é chamado *max pooling*, o que faz com que um filtro NxN atravesse toda a matriz de dados e escolha o maior elemento de cada janela N por N para ser incluído no próximo mapa de representação.

Como função de ativação dos neurônios é comumente utilizada a *Rectified Linear Unit* (ReLU), que é representada pela Equação 2.1. O cálculo da ativação e sua derivada é muito frequente, o que se faz importante o uso de uma função barata e eficiente.

$$f(x) = \max(0, x) \quad (2.1)$$

2.2.4 Redes Neurais Convolucionais

As redes neurais convolucionais usam um tipo especial de estrutura. A rede contém várias cópias do mesmo neurônio que compartilham o mesmo peso, *bias* e função de ativação. Assim, os neurônios podem ser reutilizados em diferentes partes da entrada. Eles são dispersamente conectados, ou seja, não possuem conexões densas entre todos os outros neurônios, apenas entre os vizinhos na camada anterior. (ZEILER; FERGUS, 2013)

As primeiras camadas que recebem os dados (ou sinais, à respeito de neurônios) de entrada são chamadas filtros de convolução, que é o processo de rotulação baseado no que

foi aprendido anteriormente. Ou seja, a convolução entre um sinal e um filtro de determinada característica resulta em valores mais intensos quando o sinal possui tal característica.

Para cada imagem é aplicado uma pequena janela (ou filtro) em cada conjunto de *pixels* que procura por certos recursos na imagem. O que faz sentido para dados que são espaciais e possuem características locais.

A convolução tem propriedade de ser invariante translacional, isto significa que cada filtro de convolução representa uma característica de interesse, e a *Convolutional Neural Network* (CNN) descobre quais recursos compreendem a referência resultante. A intensidade do sinal de saída não depende de onde os recursos estão localizados, mas simplesmente se os recursos estão presentes.

2.3 Transferência de Aprendizado

A combinação de modelagens muito sofisticadas e *hardware* de última geração não são suficientes para obter bons resultados em inteligência artificial. Principalmente com redes profundas, onde se tem milhões de parâmetros. É também necessária uma quantidade de dados massiva e de qualidade para obter boa generalização, o que na maioria dos casos pode ser uma tarefa próxima do impossível, visto que a aquisição de dados eleva os custos ou estes pertencem à domínios privados ou ainda não se dispõe de um grande volume de dados para o treinamento. Além disso, mesmo com o uso de *Graphics Processing Unit* (GPU) tal treinamento pode levar semanas.

A transferência de aprendizado permite lidar com tais problemas aproveitando os dados rotulados já existentes de alguma outra tarefa ou domínio relacionado. A partir de um banco de dados, certo modelo aprende diversas características que também podem ser aplicadas para outros propósitos, ou seja o conhecimento básico já existe, é apenas necessário refiná-lo. ([YOSINSKI et al., 2014](#))

Existem duas maneiras comuns para transferir conhecimento: Congela-se os pesos das camadas inferiores que foram adquiridos a partir de outra base de dados e treina-se apenas a camada densa no topo, de acordo com o novo número de classes. De outra forma, também é possível utilizar os pesos das camadas inferiores como um ponto de partida, assim treina-se a rede inteira refinando os valores das camadas escondidas e treinando a última camada superior.

De qualquer forma a Transferência de Aprendizado ainda requer que quantidades extraordinárias de dados já tenham sido treinados. Outra desvantagem é a limitação de arquitetura, visto que para utilizar conhecimento previamente adquirido é necessário que se use a mesma modelagem.

2.4 Treinamento

O treinamento de uma rede profunda de classificação tem como seu objetivo minimizar os erros entre as previsões e os rótulos reais. Para isso é preciso especificar uma função de custo que compõe os erros, e então uma função de otimização, que guie o algoritmo para cada vez mais próximo dos pesos ideais, com o menor erro entre saída produzida pela rede e a saída desejada (rótulo).

Nesta seção serão introduzidos conceitos essenciais ao treinamento de uma rede profunda, com enfoque nos métodos utilizados neste trabalho.

2.4.1 Classificação

Para classificação entre classes mutuamente exclusivas é necessário que para cada imagem haja apenas um rótulo possível. A função classificadora utilizada neste trabalho é a regressão *Softmax*. Ela recebe as pontuações geradas pela última camada de uma rede neural e normaliza esses valores transformando-os em probabilidades. Ou seja, a saída é um vetor contendo a probabilidade de cada classe estar contida na imagem. A função é denotada pela Equação 2.2.

$$S(y_i) = \frac{e^{y_i}}{\sum_{\forall j} e^{y_j}} \quad (2.2)$$

2.4.2 Função de Custo

Uma função de custo, em aprendizado de máquina, nos permite quantificar quão bom ou ruim uma devida classificação é. Existem muitas formas de calcular a diferença entre previsão e rótulo real. Neste trabalho, assim como na maioria dos trabalhos referenciados, foi utilizado entropia cruzada. A diferença entre duas distribuições de probabilidades é quantificada através do produto escalar da Equação 2.3, onde Y representa o vetor contendo os valores reais e Y' representa os valores estimados.

$$\text{Custo} = -Y \cdot \log(Y') \quad (2.3)$$

Em redes neurais, o cálculo para entropia cruzada é independente do tipo de camadas usadas e é compatível com a função *Softmax*, que garante que a distribuição das probabilidades é bem formada.

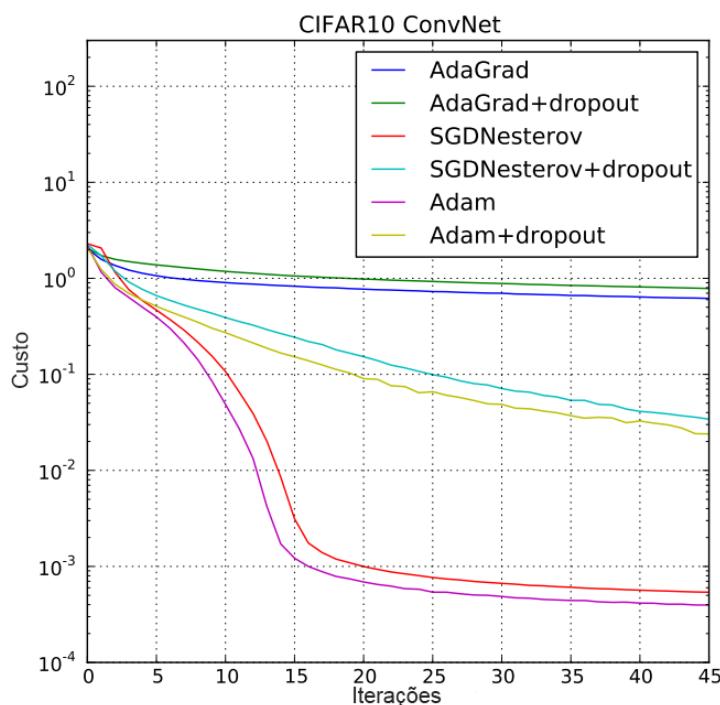
2.4.3 Otimização

Assim como a função de custo, outra decisão a ser tomada é sobre qual algoritmo de optimização seria mais adequado para minimizar a função de custo. Existem vários disponíveis e

inclusive já implementados na biblioteca *TensorFlow*. Neste trabalho foi utilizado otimização Adam, *Adaptative Moment Estimation* ([KINGMA; BA, 2014](#)), que é baseado em outros processos de otimização, AdaGrad e RMSProp, e combina as vantagens de ambos, lida com gradientes esparsos assim como ruídos.

Ao invés de apenas adaptar as taxas de aprendizado dos parâmetros com base no primeiro momento médio, Adam também faz uso da média dos segundos momentos dos gradientes (variância não centrada). O algoritmo calcula uma média móvel exponencial do gradiente e do gradiente ao quadrado, e também contra o decaimento destes, através de parâmetros constantes.

Figura 1 – Comparação entre algoritmos de otimização.



Fonte: Adaptada de [Kingma e Ba \(2014\)](#).

A Figura 1 representa uma rede convolucional treinada para reconhecimento de objetos usando diferentes técnicas de otimizações, como pode-se ver a otimização Adam leva a rede aos melhores resultados com poucas iterações.

2.5 Avaliação

Após o treinamento de uma rede profunda é necessário saber se esta realmente pode generalizar às entradas recebidas ou se está ocorrendo *overfitting*. Nesta seção, são apresentadas métricas utilizadas para avaliar o desempenho de um modelo, tanto durante treino quanto avaliação.

2.5.1 Matriz de Confusão

Uma matriz de confusão aprimora a descrição do desempenho de um modelo de classificação. Isso ajuda a visualizar e rotular a distribuição de previsões corretas e incorretas feitas pelo modelo.

Ao separar as previsões corretas e incorretas em uma matriz de confusão, pode-se identificar resultados anômalos em campos específicos. Analisando simplesmente a proporção de previsões corretas e incorretas, não é possível saber se há desequilíbrio dentro dos resultados, uma vez que uma pontuação de precisão geral alta pode ser enganosa. Na Tabela 1 a seguir, as duas previsões possíveis estão relacionadas aos dois rótulos disponíveis, dando assim um campo para todos os resultados capazes. O objetivo é que a maioria das previsões cheguem em verdadeiro positivo e verdadeiro negativo, deixando um pequeno número de falsos positivos e falsos negativos. Se houvesse um resultado anômalo, como por exemplo, uma quantidade esmagadora de falsos negativos, bem como negativos verdadeiros, isso pode não ser notado com uma divisão proporcional simples, no entanto, com uma matriz de confusão, esse problema é colocado em evidência.

Tabela 1 – Matriz de Confusão

	Previsão Cacau	Previsão Outros
Cacau	Verdadeiro Positivo	Falso Negativo
Outros	Falso Positivo	Verdadeiro Negativo

Fonte: Elaborada pelo autor.

2.5.2 Acurácia

A acurácia pode ser usada para descrever erros sistemáticos. As previsões corretas devem estar dentro de uma ampla margem de erro, porém sempre perto do valor verdadeiro. Usando a Matriz de Confusão, pode ser calculado como na Equação 2.4.

$$AC = \frac{VP + VN}{VP + VN + FP + FN} \quad (2.4)$$

2.5.3 Precisão

A precisão pode ser usada para descrever erros estatísticos. Previsões precisas terão pouca variabilidade estatística e mostrará como é possível repetir um dado resultado. No entanto, a precisão é independente da acurácia e não descreve a relação de uma previsão com um valor verdadeiro. Usando a Matriz de Confusão, pode ser calculado como na Equação 2.5.

$$PR = \frac{VP}{VP + FP} \quad (2.5)$$

2.5.4 Revocação

Revocação (sensibilidade, *recall*) é o termo usado para descrever a fração de previsões relevantes. As previsões com alta revocação são aquelas que têm uma probabilidade muito baixa de fazer um falso negativo, garantindo assim a maioria dos dados relevantes serem aprovados. Revocação não mede acurácia ou precisão. No entanto, ela pode criar resultados parciais por conta própria. Usando a Matriz de Confusão, pode ser calculado como na Equação 2.6.

$$RE = \frac{VP}{VP + FN} \quad (2.6)$$

2.5.5 Coeficiente de Correlação Matthews

Existem diversas métricas usadas para avaliar modelos de redes profundas, como acurácia, precisão e revocação. Neste trabalho foi implementado o Coeficiente de Correlação Matthews (MCC) porque este leva todos os demais em consideração e é o mais indicado para classificações binárias.

Pode-se calcular o valor MCC diretamente da matriz de confusão como na Equação 2.7.

$$MCC = \frac{VP.VN - FP.FN}{\sqrt{(VP + FP).(VP + FN).(VN + FP).(VN + FN)}} \quad (2.7)$$

$$-1 < MCC < 1 \quad (2.8)$$

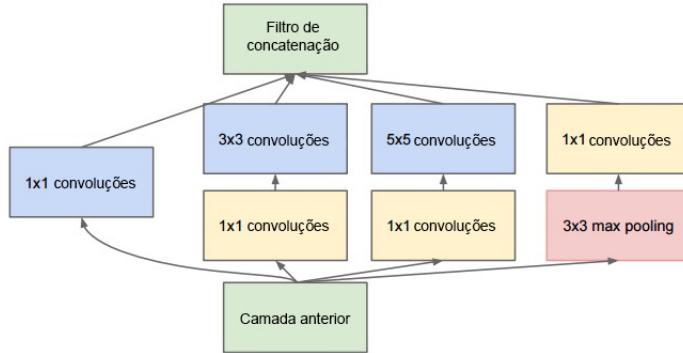
Na Equação 2.8 podem ser encontrados os limites deste coeficiente. Quando $MCC = -1$ há perfeito desacordo entre a verdade e a previsão, $MCC = 1$ há perfeita concordância e $MCC = 0$ significa que a previsão é aleatória em relação à verdade.

2.6 Modelos

2.6.1 Inception

Diferentemente dos modelos de redes neurais anteriores, o módulo *Inception* utiliza múltiplos filtros de convolução processados na mesma entrada. Ao invés de ser escolhido um filtro apropriado para o tipo de dado, o que é longe de trivial, neste modelo são aplicados três filtros (5x5, 3x3 e 1x1), veja a Figura 2. Não somente, também é adicionado um bloco *max pooling*, dado que este é essencial para o sucesso das redes convolucionais. Todos os resultados são então concatenados, permitindo que o modelo extraia tanto características gerais (filtro 5x5) quanto locais (filtro 1x1).

O problema relacionado a esta ideia é que mesmo com um número pequeno de filtros 5x5 o custo computacional se torna excessivo, sobretudo quando unidades *pooling* são adicionadas,

Figura 2 – Módulo *Inception*.

Fonte: Adaptada de [Szegedy et al. \(2014\)](#).

pois o número de filtros na saída é o mesmo que o número de filtros da etapa anterior. Então a cada camada os resultados seriam inevitavelmente maiores tornando este modelo ineficiente. Para resolver esta explosão computacional é preciso reduzir as dimensões. Por isso aplica-se um filtro 1x1 em conjunto com ativação ReLU.

Tendo como principal elemento os módulos *Inception*, pesquisadores criaram a rede *GoogleLeNet* com 22 camadas de profundidade. Tal rede foi a vencedora da famosa competição *ImageNet* em 2014 ([SZEGEDY et al., 2014](#)).

2.6.2 Redes Residuais

Conhecer a profundidade apropriada para uma rede neural é longe de trivial. Caso esta seja profunda demais os erros não são propagados de volta corretamente. Caso sejam consideradas poucas camadas, algumas características podem ser não suficientemente identificadas.

Há também o risco de ocorrer degradação, como citado em ([HE et al., 2015](#)). Onde foi observado que, a partir de determinado ponto, quando adicionadas mais camadas, a perda e a acurácia pioravam. Isso porque neste processamento extra os pesos podem ser degradados, o que leva a uma significante perda de desempenho.

As Redes Residuais trazem uma solução para isto. Elas garantem que a cada camada o modelo possa apenas agregar aprendizado. Isto ocorre porque, no pior caso, os blocos dessas camadas atuam como um mapeamento de identidade, apenas propagando as características aprendidas anteriormente ou aperfeiçoando-as. Ou seja, a entrada de uma camada é adicionada à sua saída.

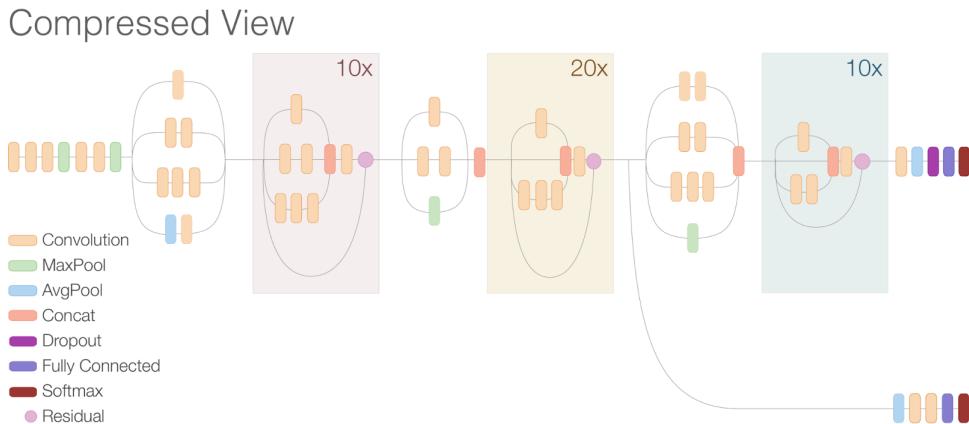
2.6.3 Inception-Resnet-v2

2.6.3.1 Arquitetura

Este modelo combina conexões residuais e versões aperfeiçoadas dos módulos *Inception*. Apesar do alto incremento em custo computacional, há ganhos significativos em reconhecimento.

Esta rede é extremamente profunda e larga, possui 185 blocos convolucionais divididos entre suas 50 camadas e apenas um bloco completamente denso. A Figura 3 mostra uma simplificação, onde os módulos híbridos (*Inception residual*) foram comprimidos para facilitar a visualização.

Figura 3 – Rede *Inception-Resnet-v2*.



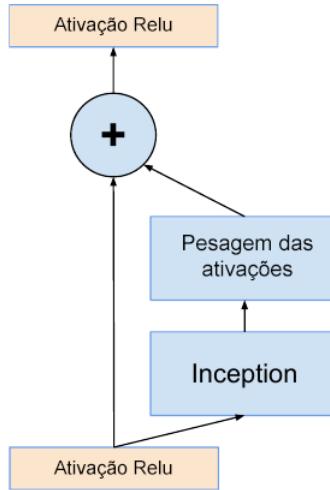
Fonte: Adaptada de [Alemi \(2016\)](#).

Nos blocos híbridos, foi substituída a etapa de concatenação dos resultados pela conexão residual (ou mapeamento de identidade), o que permite este modelo agregar os benefícios das Redes Residuais e ainda assim manter certa eficiência computacional. Foi também adicionado um filtro convolucional de 1x1 para aumentar a dimensão que é diminuída no interior do módulo, fazendo com que, deste modo, os formatos de entrada e saída sejam iguais e os blocos empilháveis.

Apesar da normalização de lote ser altamente vantajosa, neste modelo esta foi implementada apenas nas camadas tradicionais, dado que caso tal técnica fosse amplamente utilizada o custo computacional seria exacerbado.

Em ([SZEGEDY; IOFFE; VANHOUCKE, 2016](#)), assim como em ([HE et al., 2015](#)), foi observado que quando o número de filtros ultrapassa os milhares os resíduos, geram instabilidades e fazem com que a rede produza apenas zeros após poucas interações. A solução encontrada para o modelo *Inception-ResNet* foi de aplicar um peso entre 0,1 e 0,3 aos resíduos antes destes serem somados (Veja Figura 4).

Figura 4 – Esquema usado para aplicar peso aos resíduos.



Fonte: Adaptada de [Szegedy, Ioffe e Vanhoucke \(2016\)](#).

2.6.3.2 Banco de Dados do Pré-Treino

Em agosto de 2016, o modelo *Inception-ResNet-v2* ([ALEMI, 2016](#)) foi disponibilizado pela *Google* afim de estimular ainda mais progressos na área de reconhecimento de imagens. Além disso, também foi disponibilizado o ponto de inspeção (*checkpoint*) treinado no banco de dados *ImageNet* 2012 ([IMAGENET, 2012](#)).

ImageNet ([RUSSAKOVSKY et al., 2015](#)) é um projeto contínuo para facilitar aos pesquisadores acesso a um grande banco de dados de imagens. No total, existem mais de 10 milhões de imagens representando mais de 10 mil categorias. Todos os anos acontece o desafio *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC). No qual é apresentado o estado da arte da área e testados os melhores modelos de aprendizado de máquina. Em 2012 ([KRIZHEVSKY; SUTSKEVER; HINTON, 2012](#)) apresentou a arquitetura vencedora *AlexNet*, uma das primeiras a utilizar redes convolucionais profundas.

O conjunto de dados utilizado para pré-treino deste trabalho possui 1,2 milhões de imagens, com mil categorias e mais 150 mil imagens para teste e validação. Dentre as categorias não há apenas plantas e frutas, mas também diversos objetos, animais, comidas e até roupas.

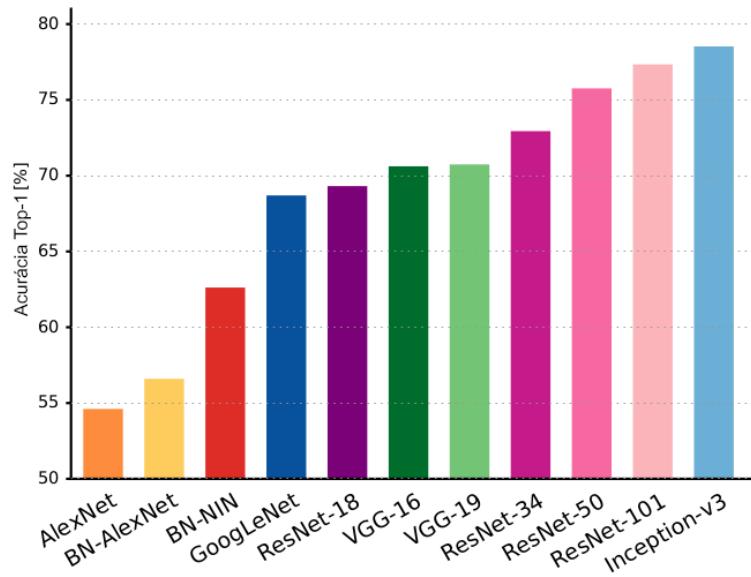
2.6.3.3 Comparações

De acordo com ([SZEGEDY; IOFFE; VANHOUCKE, 2016](#)), todas as versões residuais do modelo Inception obtém não apenas melhores resultados como também chegam a estes mais rapidamente que suas versões puras.

A comparação de arquiteturas feita em ([CANZIANI; PASZKE; CULURCIELLO, 2016](#)) mostra que as redes Inception-v3 e Inception-v4 superam todos os outros modelos analisados

por pelo menos 7% em acurácia. Veja a Figura 5.

Figura 5 – Resultados Top-1 por rede.



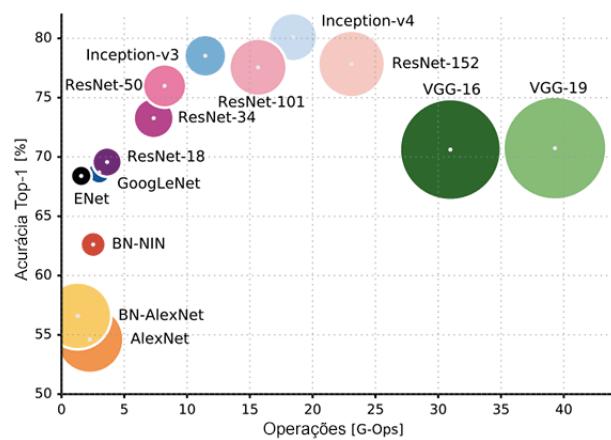
Fonte: Adaptada de [Canziani, Paszke e Culurciello \(2016\)](#).

Ainda comparando acurácia por operações e quantidade de parâmetros, ([CANZIANI; PASZKE; CULURIELLO, 2016](#)) mostra que os modelos que apresentam melhores resultados são Inception-v3, Inception-v4 e as variações densas de ResNet, fazendo redes que são amplamente usadas nos dias atuais, como VGG e AlexNet, parecerem desinteressantes, como pode ser visto na Figura 6. Pode-se concluir que caso Inception-Resnet-v2 fosse agregado ao estudo, este teria obtido resultados ainda melhores, visto que além de sua arquitetura ser baseada no modelo Inception-v4, as conexões residuais incorporam apenas ganho em acurácia em troca de pouco custo computacional.

2.7 Considerações Finais

Neste capítulo discorreu-se sobre modelagens de redes profundas, suas métricas para treinamento e avaliação, dentre outras terminologias úteis para o entendimento deste trabalho. No capítulo a seguir será discutido o desenvolvimento e resultados obtidos.

Figura 6 – Comparação entre acurácia, operações e quantidade de parâmetros. O tamanho da circunferência é proporcional ao número de parâmetros da rede.



Fonte: Adaptada de [Canziani, Paszke e Culurciello \(2016\)](#).

Capítulo 3

DESENVOLVIMENTO

3.1 Considerações Iniciais

Neste capítulo são discutidas as particularidades do desenvolvimento deste trabalho, do banco de dados, e mostradas comparações e análises entre diferentes tipos de aplicações de transferência de aprendizado. Por fim, serão descritas as dificuldades e limitações encontradas durante a produção destes resultados.

3.2 Descrição da Pesquisa Proposta

Devido à dificuldade em encontrar um conjunto de dados suficientemente grande, foi decidido utilizar as técnicas de transferência de aprendizado que são apropriadas para pequenos conjuntos de dados, como é o caso da base de dados disponível contendo imagens de cacau.

A rede *Inception-ResNet-v2* foi escolhida por ter alto desempenho, custo computacional razoável, código aberto através da sub biblioteca *TF-slim* e ponto de inspeção disponível treinado em uma das maiores bases de dados rotulados (*ImageNet*).

Durante este capítulo serão discutidos os três tipos de treinamentos realizados:

- **Treino total:** Treinamento de todas as camadas usando os pesos do modelo de inspeção como ponto de partida.
- **Treino parcial:** Congelamento dos pesos da maioria das camadas escondidas, treinamento da última redução convolucional e camadas superiores (*pooling*, *dropout* e camada densa de saída).
- **Treino mínimo:** Congelamento de todas as camadas escondidas, treinamento apenas nas camadas superiores (*pooling*, *dropout* e camada densa de saída).

3.3 Materiais

Os treinamentos foram processados em uma *GPU GeForce GTX 1060* com 3GB de memória, 1728 núcleos CUDA, velocidade 1771,5 MHz e capacidade para até 18.432 processos

concorrentes, está máquina pertence ao Laboratório de Aprendizado de Máquina do ICMC-USP. Foi também utilizado a *Central Processing Unit* (CPU) para loteamento dos dados, o que aliviou levemente a carga de processamento da placa gráfica e tornou cada passo 0,01 segundo mais rápido.

Foi utilizada a biblioteca de código aberto *TensorFlow* e sua sub biblioteca *TF-Slim*, que contém abstrações de modelos, arquiteturas e também facilita a codificação e decodificação dos bancos de dados para o formato *tfrecord*.

3.4 Base de Dados

O maior revés que um projeto de *Deep Learning* enfrenta é a falta de dados confiáveis, excepcionalmente para um projeto focado em um campo que não tenha sido previamente abordado, onde nenhum dado foi previamente coletado e aberto para uso público. Para treinar um modelo preciso, uma grande quantidade de dados variados deve ser coletada. Não é por acaso que as maiores empresas de propagação de dados do mundo tenham os resultados mais bem-sucedidos dos seus projetos na área de aprendizagem de máquinas, pois têm acesso ilimitado aos dados necessários para o sucesso.

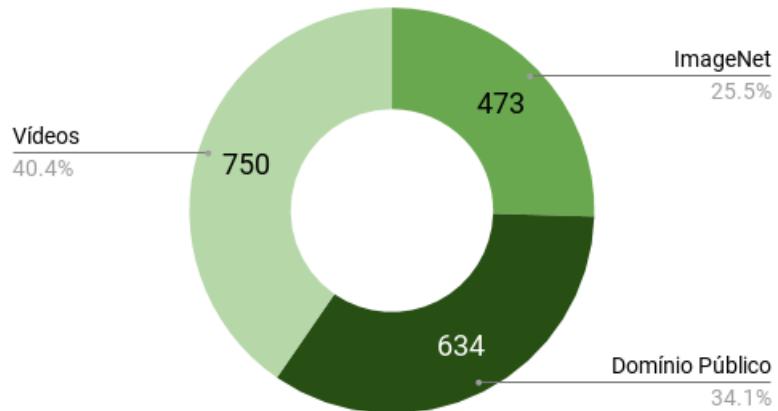
Devido à falta de bancos de dados pré-existentes, uma coleção de imagens de alta qualidade teve de ser construída a partir do zero. Para treinar um modelo confiável para a identificação dos frutos de cacau, é necessário que haja imagens suficientes que os mostrem em cada estágio de desenvolvimento (brotos, verdes, não maduros e maduros), em tantos ângulos diferentes quanto possível, e em uma variedade de configurações. Trabalhando dentro das limitações discutidas, várias fontes foram usadas para garantir uma variação de quantidade viável.

Um banco de dados contendo 1857 imagens de frutos de cacau e 1857 imagens de outras plantas e frutas foi coletado de várias fontes. A grande base ImageNet forneceu uma boa proporção, e muitos outros exemplos de alta qualidade foram extraídos de várias páginas em domínio público Figura 7.

Os fotógrafos que disponibilizam seu trabalho para o domínio público apenas tendem a fornecer imagens centradas e bem compostas de seus objetos. Devido a isso, existe o risco de o banco de dados ter exemplos insuficientes de frutos de cacau em ângulos diferentes, o que pode criar uma polarização dentro do sistema.

Vídeos são fontes facilmente alcançáveis com números elevados de imagens em vários ângulos e distâncias diferentes. Um vídeo digital padrão é executado em 24 quadros por segundo e com resolução variando entre 420p até 1080p. Supondo que o vídeo seja panorâmico ou alterando o ângulo, isso significa que um vídeo de 1 minuto poderia fornecer até 1440 imagens estáticas de um fruto de cacau em diferentes pontos de vista.

Figura 7 – Fonte das imagens que compõem o conjunto de dados.



Fonte: Elaborada pelo autor.

Usando um *software* de edição de imagem gráfica, foram importados vários vídeos provenientes de domínio público, e então os quadros foram divididos e renderizados como imagens estáticas. Cada sexto quadro foi extraído, de modo a garantir que cada imagem tenha uma mudança em ângulo suficientemente grande (Figura 8). Após a importação, os quadros que não continham cacau foram excluídos, durante o qual foi feita uma garantia de qualidade, antes que os *scripts* fossem usados para cortar e salvar automaticamente as imagens no banco de dados.

Figura 8 – Quadros-chave de um vídeo sobre frutos de cacau para garantir um equilíbrio de variedade.



Fonte: Elaborada pelo autor.

Os melhores exemplos para mostrar ao modelo de imagens sem cacau são outras formas de flora. À medida que o modelo se torna mais sofisticado, é adequado garantir que ele não associe as características dos frutos de cacau à de outras árvores frutíferas similares. Tendo isso em vista, foi utilizado um grande número de frutas que compartilham características básicas com o cacau. Muitas imagens de frutas amarelas e vermelhas, bem como frutas que compartilham qualidades físicas semelhantes foram fornecidas. Das 1857 imagens utilizadas, 1113 foram obtidas a partir da ([IMAGECLEF, 2015](#)), uma organização dedicada à coleta e preservação de imagens de diversos perfis de flora e fauna, com o objetivo de explorar análises visuais. As restantes 744 imagens foram colhidas a partir de várias fontes de domínio público, incluindo a *ImageNet*, especificamente para suas semelhanças com o fruto do cacau, conforme discutido.

Durante o treino as imagens são pré-processadas e distorcidas, o que é útil para aumentar o conjunto de dados e tornar a rede invariante aos aspectos das imagens que não afetam seu rótulo. As distorções consistem de (1) redimensionamento¹ para o tamanho estático de 299x299 *pixels*, (2) inversão horizontal aleatória, (3) distorção aleatória das cores. Para a avaliação, as imagens são apenas redimensionadas para 299x299 *pixels*.

A Figura 9 contém exemplos positivos (com cacau) de imagens que compõem o banco de dados deste trabalho. É possível perceber a variedade tanto de cores como de ângulos.

Figura 9 – Exemplos de imagens com cacau que compõem o banco de dados deste trabalho.



Fonte: Elaborada pelo autor.

3.5 Descrição das Atividades Realizadas

A primeira parte a ser tratada durante a implementação deste trabalho foi o loteamento dos dados. Após estes terem sido decodificados de JPG para o formato tfrecord, foi gerado um *tf.Tensor* que despacha os lotes durante o treinamento. Neste foram configurados 4 *threads* concorrentes para transporte dos dados do disco rígido para a memória da placa de vídeo. Caso mais processos fossem utilizados, o treinamento poderia ser mais rápido, porém utilizando-se essas configurações o sistema já se encontrava no limite da capacidade de memória. O loteamento seleciona imagens aleatoriamente do banco de dados, portanto mesmo com muitas iterações é possível ter lotes diferentes para cada passo.

O tamanho de lote também sofreu com o limitante de memória, o maior número possível configurável foi de 20 imagens para o treino parcial e mínimo, o que levou a média do custo a atingir variações drásticas durante treinamento. Caso a quantidade de dados pudesse ser maior, o modelo teria gerado gradientes mais estáveis, o que possivelmente levaria o modelo a convergir mais rápido.

Para treinamento do modelo foi usado o ponto de inspeção fornecido em (ALEMI, 2016), que traz os pesos de todas as 50 camadas e 185 blocos convolucionais. O modelo *InceptionResnet-v2* e este ponto de inspeção foram, então, importados através do código confeccionado para este

¹ Isso pode deformar a relação de aspecto das imagens caso estas não sejam quadradas, todavia isto não influencia significativamente nos resultados.

trabalho. Por motivo de o modelo original ter sido treinado para classificação de 1000 classes, foi preciso descartar os pesos da camada densa superior e modificar o grafo do módulo em vista de acomodar saídas binárias.

Como função de custo foi usada a função `tf.losses.softmax_cross_entropy`, que recebe como parâmetro o tensor da camada superior e internamente aplica *Softmax* e calcula a entropia cruzada.

A otimização Adam é a mais indicada para a grande maioria de aplicações de redes convolucionais (KINGMA; BA, 2014) e a melhor disponível na biblioteca *Tensorflow*. Em conjunto foi aplicado uma taxa de aprendizado com decaimento exponencial. Sendo um dos hiper-parâmetros mais significativos, esta foi largamente experimentada. O maior valor máximo possível ao início dos treinos foi de 0,001.

Durante o treinamento também foi aplicado uma taxa de 80% de *Dropout*, que é uma técnica de normalização e prevenção de *overfitting* comum para redes neurais, onde esta porcentagem representa a probabilidade de um neurônio ser mantido, ou seja os outros serão esquecidos, o que estimula os pesos a atingirem resultados através de diferentes conexões (SRIVASTAVA *et al.*, 2014).

Devido à natureza altamente modular que a biblioteca *Tensorflow* provê aos modelos que o usam, é possível facilmente selecionar quais camadas treinar e quais "congelar", portanto a profundidade do treinamento também foi um parâmetro experimentado. Contudo, neste documento serão descritos apenas os mais pertinentes, que são treino total, parcial e mínimo.

Durante treinamento as métricas e pesos são salvas a cada 10 passos e pontos de inspeção ao final de cada iteração. Para esclarecimento, uma iteração significa uma passagem por todos os dados do conjunto.

3.6 Resultados

Nesta subseção serão comparados os resultados obtidos com os três treinos propostos em 136 iterações, que tiveram duração indicada na Tabela 2 para cada tipo de treino.

Tabela 2 – Tempo de treinamento.

Treino total	9h33min
Treino parcial	2h19min
Treino mínimo	2h13min

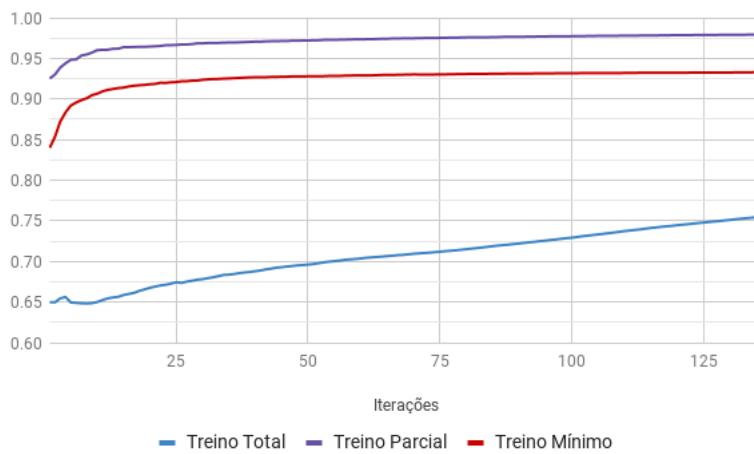
Fonte: Elaborada pelo autor.

É importante destacar que durante o treino total foi possível utilizar lotes de apenas 3 imagens, visto que a carga computacional aumenta extremamente quando é treinado o modelo completo. Seria possível diminuir a dimensão das imagens e alocar mais dados por passo,

porém esse processo também prejudica o treinamento, visto que imagens redimensionadas para tamanhos muito pequenos perdem características, o que não é desejável quando o modelo é profundo e sofisticado. Como mencionado anteriormente, o tamanho pequeno do lote tem influência negativa na velocidade de convergência, no entanto, este problema pode ser suavizado com uma taxa de aprendizado suficientemente pequena.

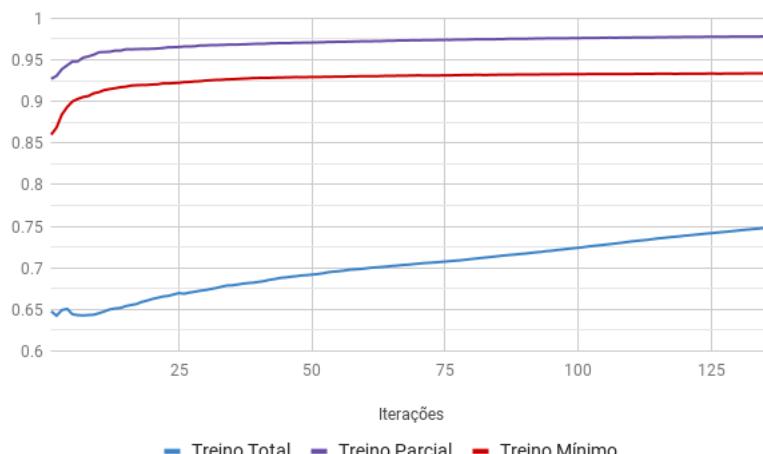
Nas Figuras 10, 11 e 12 pode-se visualizar através das iterações o desenvolvimento da acurácia, precisão e revocação, respectivamente. As curvaturas das três métricas são similares à curvatura do coeficiente MCC (Figura 13).

Figura 10 – Gráfico da acurácia por iterações.



Fonte: Elaborada pelo autor.

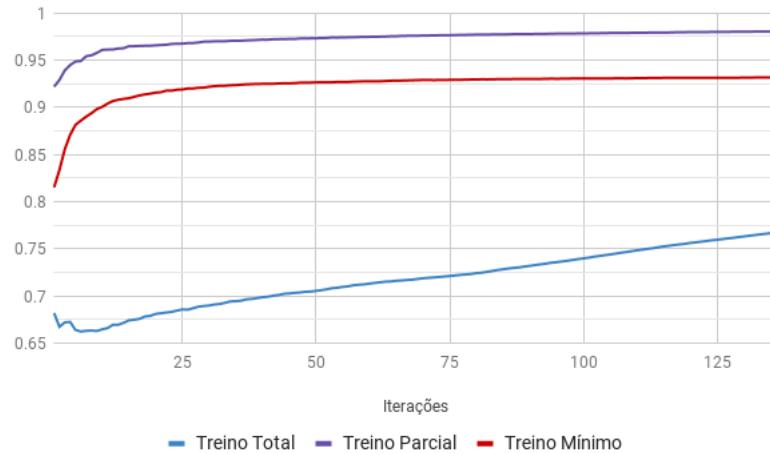
Figura 11 – Gráfico da precisão por iterações.



Fonte: Elaborada pelo autor.

É interessante perceber na Figura 13 que mesmo com um lote relativamente pequeno (20 imagens), os treinos parciais e mínimos convergem para 0,93 e 0,84, respectivamente, já nas

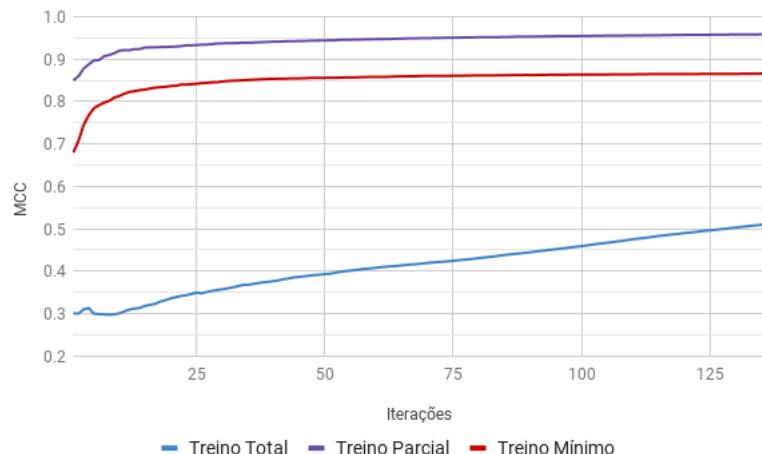
Figura 12 – Gráfico da revocação por iterações.



Fonte: Elaborada pelo autor.

primeiras 25 iterações, enquanto o treino total possui $MCC = 0,34$.

Figura 13 – Gráfico do coeficiente MCC por iterações.

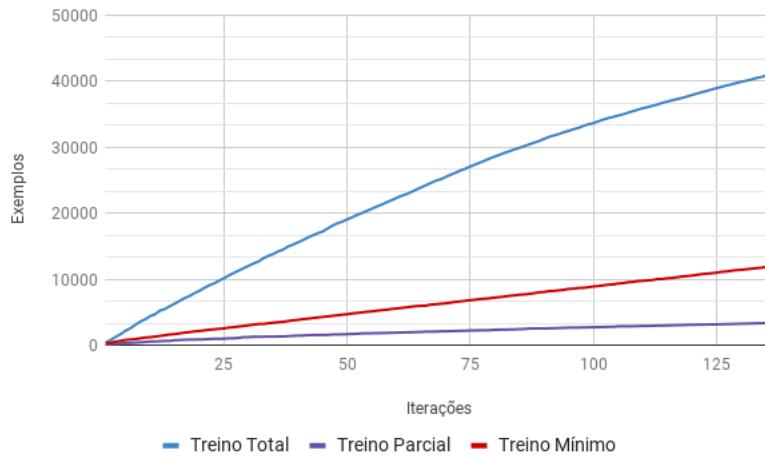


Fonte: Elaborada pelo autor.

Algo menos esperado é visto nas Figura 14 e Figura 15, que mostra a quantidade total de falsos negativos e falsos positivos. Seria intuitivo pensar que como o treino parcial e o treino mínimo têm tamanhos muito semelhantes, as curvas das previsões falsas também seriam muito semelhantes, porém como é mostrado neste experimento, a estreita camada convolucional que os diferencia é suficiente para uma drástica mudança nos resultados. As Figura 16 e Figura 17 retratam as quantidades de verdadeiros negativos e positivos.

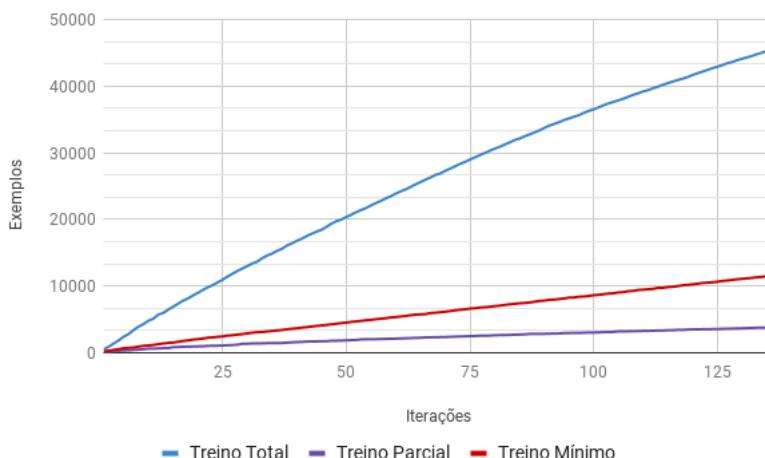
É interessante visualizar nas Figura 18 e Figura 19, as distribuições das probabilidades para o treino parcial e para o total. Como esperado, no treino parcial, junto com o incremento das iterações, as probabilidades “incertas” vão ficando mais esparsas, até que, ao fim do treino, o

Figura 14 – Soma da quantidade de falsos negativos durante o treinamento.



Fonte: Elaborada pelo autor.

Figura 15 – Soma da quantidade de falsos positivos durante o treinamento.



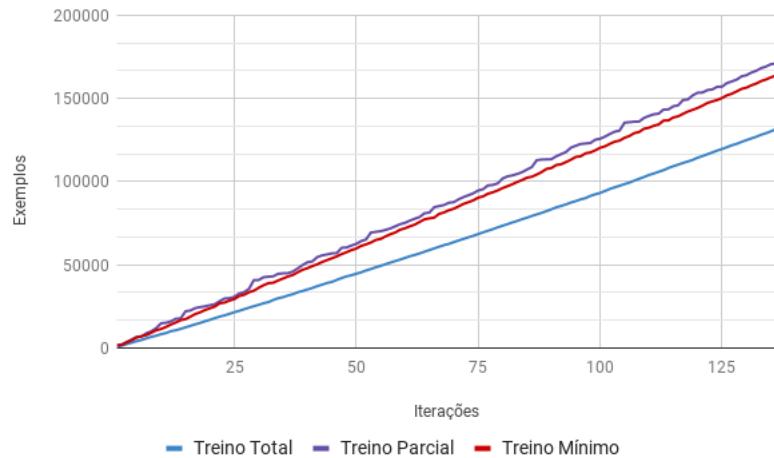
Fonte: Elaborada pelo autor.

modelo tem todas as suas previsões com alta probabilidade.

Para destacar a importância da transferência de aprendizado também foi feito um experimento de treino total sem utilizar o ponto de inspeção fornecido em conjunto com o modelo *Inception-Resnet-v2*. A Figura 20 mostra que a qualidade dos resultados do treino sem o ponto de inspeção começa a aumentar depois de 100 iterações e que existe potencial para resultados razoáveis, porém, o treino a partir do ponto de inspeção é consideravelmente mais promissor.

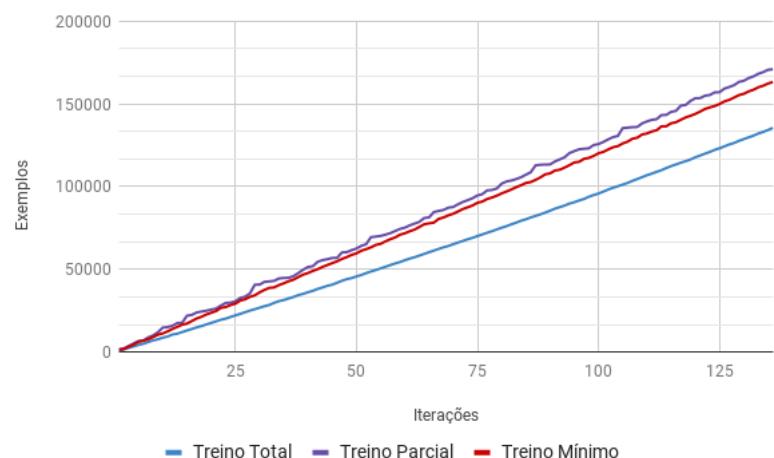
O treino parcial gera um desempenho ligeiramente inferior durante avaliação, entretanto, tal deterioração é suficientemente pequena e pode-se considerar que o modelo não sofre de *overfitting*. A Tabela 3 faz a comparação entre os resultados observados durante treinamento e avaliação.

Figura 16 – Soma da quantidade de verdadeiros negativos durante o treinamento.



Fonte: Elaborada pelo autor.

Figura 17 – Soma da quantidade de verdadeiros positivos durante o treinamento.



Fonte: Elaborada pelo autor.

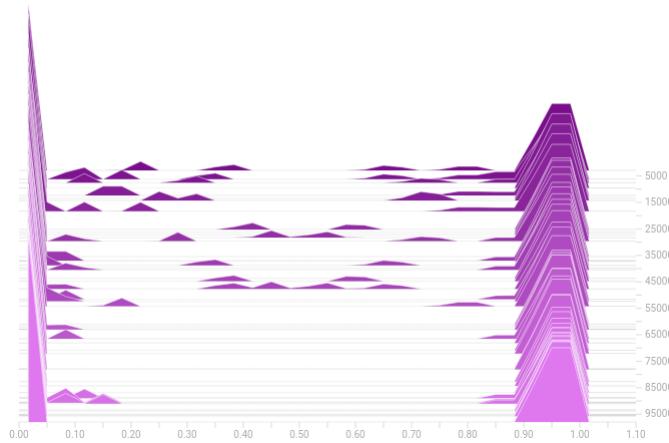
Tabela 3 – Comparação do coeficiente MCC durante treino e avaliação.

Métricas	Treino total - 136 ite.	Treino parcial - 376 ite.	Treino mínimo - 759 ite.
Treinamento	0,5108	0,9814	0,8739
Avaliação	0	0,9687	0,9314

Fonte: Elaborada pelo autor.

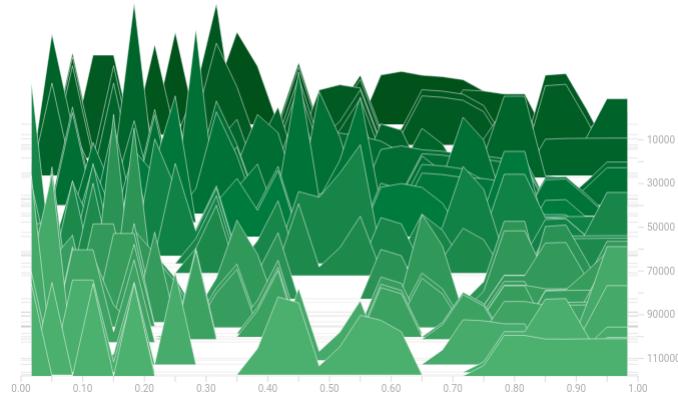
No Appendix A o leitor é convidado a visualizar alguns resultados gerados pelo modelo do treino parcial em imagens que não pertencem ao banco de dados, portanto não foram usadas nem para treino nem para avaliação.

Figura 18 – Representação espacial das previsões durante o treino parcial. No eixo y encontram-se as quantidades de exemplos, no eixo x as probabilidades e no eixo z os passos do treinamento



Fonte: Elaborada pelo autor.

Figura 19 – Representação espacial das previsões durante o treino total. No eixo y encontram-se as quantidades de exemplos, no eixo x as probabilidades e no eixo z os passos do treinamento

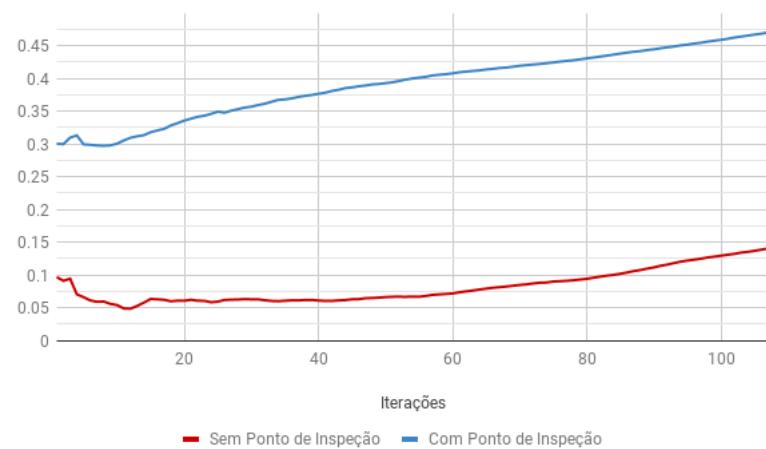


Fonte: Elaborada pelo autor.

3.7 Dificuldades e Limitações

As duas maiores dificuldades encontradas para este trabalho são também as dificuldades da maioria de implementações em *Deep Learning*. A disponibilidade de recursos computacionais foi um grande gargalo durante as experimentações, como visto anteriormente, o tipo de treino que exige a menor quantidade de recursos requer, ao menos, duas horas para atingir 100 iterações, o que multiplicado à falta de técnicas matemáticas para escolha de ótimos hiper-parâmetros, significa numerosas tentativas. Outro fator limitante foi aquisição de dados. Mesmo as eficientes técnicas de conversão de vídeos para imagens estáticas ainda requerem quantidades significantes de tempo.

Figura 20 – Comparação do coeficiente MCC durante o treino total com e sem o ponto de inspeção.



Fonte: Elaborada pelo autor.

Capítulo 4

CONCLUSÃO

4.1 Contribuições

Os resultados fornecidos mostram uma habilidade mais do que satisfatória para identificar pastilhas de cacau em imagens estáticas. Com um MCC de 0,96, o modelo já pode, no seu estado atual, de forma confiável, ser usado para fins de identificação. Os resultados também demonstraram como a transferência de aprendizado tem a possibilidade de permitir que experimentos como este prosperem, mesmo com pequenos conjuntos de dados à sua disposição. Com um modelo *Deep Learning* sofisticado de código aberto disponível, com pontos de verificação treinados em tipos de dados semelhantes, a transferência de aprendizagem pode se tornar um catalisador, impulsionando pequenas expansões na área de Aprendizado de Máquinas para maiores níveis de qualidade, como este. Como a falta de grandes quantidades de dados sempre foi uma obstrução predominante no aprendizado da máquina, a transferência de aprendizagem permite "posicionar-se sobre os ombros dos gigantes".

4.2 Relacionamento entre o Curso e o Projeto

O curso de Engenharia de Computação não possui matérias obrigatórias específicas sobre aprendizado de máquina, o mais próximo é Inteligência Artificial, que cobre vagamente aspectos desenvolvidos neste projeto. Todavia, é indiscutível a importância da sólida base matemática que o curso proporciona. Estatística, Cálculos, Equações Diferenciais Ordinárias e Cálculo Numérico são disciplinas que foram essenciais para o entendimento e desenvolvimento do tema deste trabalho.

Diversas outras disciplinas, apesar de não serem diretamente relacionadas, também contribuíram para que a autora soubesse como e onde adquirir conhecimento. Além de proporcionar treinamento em programação de diversas linguagens, que garantiram agilidade para compreender novas bibliotecas, como o *TensorFlow*.

4.3 Considerações sobre o Curso de Graduação

De maneira geral, a autora possui uma visão extremamente positiva em relação ao curso. Que pode ser acentuado pelos seguintes pontos: Diversos docentes bem preparados e

com currículos exemplares, ótima infraestrutura de laboratórios, muitas oportunidades para participação em atividades extra-curriculares.

Dentre os pontos fracos da graduação está a variação da importância dada a certas disciplinas. Algumas são tratadas como algo secundário tanto por discentes quanto por docentes, que deixam a desejar em suas metodologias de ensino, e outras, no entanto, extrapolam os créditos planejados de tal forma que não deixam espaço para dedicação em outras esferas. Essas diferenças apenas atrapalham um jovem engenheiro a conhecer e apreciar todos os caminhos que a universidade deveria mostrar.

Em muitas ocasiões, o curso de Engenharia de Computação pode ser maçante e pesado, o que em conjunto com a falta de disciplinas práticas e motivantes nos primeiros semestres leva à alta evasão de alunos. Porém, nos últimos anos o curso ganhou importantes mudanças na grade curricular que melhoraram a forma como os alunos adquirem intimidade tanto com o ICMC quanto com o Departamento de Engenharia Elétrica, e assim, desenvolvem conhecimento na área.

4.4 Trabalhos Futuros

Apesar do sucesso deste trabalho, ainda há grandes passos a serem feitos para chegar-se ao produto acabado. Atualmente, está comprovado que um fruto de cacau pode ser identificado, em qualquer estágio de desenvolvimento e contra frutas e objetos visualmente muito semelhantes. A combinação de transferência de aprendizagem e o conjunto de dados razoável tornou isso possível. No entanto, o próximo estágio de desenvolvimento deve ser expandir o modelo, ensinando-o não só a identificar frutos de cacau, mas também classificar se está maduro e pronto para ser colhido. Para conseguir isso, o conjunto de dados deve ser expandido, com exemplos suficientes e equilibrados de cada estágio de desenvolvimento, garantindo que uma linha clara entre os dois estados possa ser desenhada e distinguida.

Uma vez que um modelo forte é estabelecido, ele precisará ser otimizado para permitir a identificação em tempo real. Um veículo terrestre seria capaz de parar e concentrar-se em alvos potenciais, permitindo que o modelo tenha tempo suficiente para fazer uma identificação bem sucedida, enquanto um *drone* estará em constante movimento e muito pouco provável que seja completamente estacionário em qualquer ponto. Para tornar isso possível, o modelo deve ser otimizado para que ele possa reconhecer um fruto de cacau maduro quase que instantaneamente, além de compensar o movimento constante da câmera. Uma alternativa para isso seria adaptar o modelo a outro pré-existente em tempo real, como o YOLO (REDMON *et al.*, 2015) que apresentou muito potencial no campo de detecção de objetos em tempo real.

Com uma identificação rápida e altamente precisa estabelecida, continua a existir a programação comportamental do próprio *drone*. É altamente provável que a detecção de colisões tenha de ser adaptada para adequar-se aos perigos colocados dentro de uma plantação de cacau.

Além disso, o *drone* precisará ter equipamentos especializados para cortar o fruto da árvore sem danificá-lo, o que também pode exigir treinamento adicional separado no modelo se não houver solução mecânica disponível.

Uma vez que o *drone* é considerado viável em aspectos de *software* e *hardware*, ainda haveria a questão da interação com o usuário. Um aplicativo de ponta precisaria ser construído para permitir que os trabalhadores das plantações interajam com o dispositivo e planejem rotas, estabeleçam limiares e obtenham acesso a estatísticas.

Quando todas estas etapas estiverem concluídas, haverá um grande potencial para que o modelo seja adaptado, usando transferência de aprendizagem, mais uma vez, para identificar outras frutas. Considerando isto, prova-se que projetos de aprendizado de máquinas, únicos e especializados, podem dar origem a revoluções em todo um campo de automação. Se todos os trabalhos futuros mencionados forem bem-sucedidos em combinação, é possível finalmente trazer variadas formas de colheitas, uma vez deixadas para trás, na era moderna da automação.

REFERÊNCIAS

- ALEMI, A. **Improving Inception and Image Classification in TensorFlow**. 2016. Disponível em: <<https://research.googleblog.com/2016/08/improving-inception-and-image.html>>. Acesso em: 18 ago. 2017. Citado 3 vezes nas páginas 32, 33 e 40.
- CANZIANI, A.; PASZKE, A.; CULURCIELLO, E. An analysis of deep neural network models for practical applications. **CoRR**, abs/1605.07678, 2016. Disponível em: <<http://arxiv.org/abs/1605.07678>>. Citado 3 vezes nas páginas 33, 34 e 35.
- HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. New York: Macmillan, 1994. Citado na página 24.
- HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. **CoRR**, abs/1512.03385, 2015. Disponível em: <<http://arxiv.org/abs/1512.03385>>. Citado 2 vezes nas páginas 31 e 32.
- IMAGECLEF. **LifeCLEF 2015 Plant task**. 2015. Disponível em: <<http://www.imageclef.org/lifeclef/2015/plant>>. Acesso em: 18 ago. 2017. Citado na página 39.
- IMAGENET. **Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)**. 2012. Disponível em: <<http://image-net.org/challenges/LSVRC/2012/>>. Acesso em: 18 ago. 2017. Citado na página 33.
- KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **CoRR**, abs/1412.6980, 2014. Disponível em: <<http://arxiv.org/abs/1412.6980>>. Citado 2 vezes nas páginas 28 e 41.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <<http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>>. Citado na página 33.
- REDMON, J.; DIVVALA, S. K.; GIRSHICK, R. B.; FARHADI, A. You only look once: Unified, real-time object detection. **CoRR**, abs/1506.02640, 2015. Disponível em: <<http://arxiv.org/abs/1506.02640>>. Citado na página 50.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. In: RUMELHART, D. E.; MCCLELLAND, J. L. (Ed.). **Parallel Distributed Processing**. [S.l.]: MIT Press, 1986. v. 1, p. 318–362. Citado na página 24.
- RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015. Citado na página 33.

SRIVASTAVA, N.; HINTON, G. E.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, n. 1, p. 1929–1958, 2014. Disponível em: <<http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf>>. Citado na página 41.

SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V. Inception-v4, inception-resnet and the impact of residual connections on learning. **CoRR**, abs/1602.07261, 2016. Disponível em: <<http://arxiv.org/abs/1602.07261>>. Citado 2 vezes nas páginas 32 e 33.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. E.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. **CoRR**, abs/1409.4842, 2014. Disponível em: <<http://arxiv.org/abs/1409.4842>>. Citado na página 31.

YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H. How transferable are features in deep neural networks? **CoRR**, abs/1411.1792, 2014. Disponível em: <<http://dblp.uni-trier.de/db/journals/corr/corr1411.html#YosinskiCBL14>>. Citado na página 26.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. **CoRR**, abs/1311.2901, 2013. Disponível em: <<http://arxiv.org/abs/1311.2901>>. Citado na página 25.

APÊNDICE A

EXEMPLOS E PROBABILIDADES

Figura 21 – Ausência de cacau: 90,96%



Figura 22 – Ausência de cacau: 99,22%



Figura 23 – Presença de cacau: 97,91%



Figura 24 – Presença de cacau: 92,81%

