GUSTAVO DONNINI CHEN
PAULO ROBERTO MACHADO DOS SANTOS
DIEGO HIDEK CAETANO IDE

AUTOMAÇÃO DO PROCESSO DE REFINO DA MASSA DE CACAU - AMAZÔNIA 4.0

São Paulo
2022

GUSTAVO DONNINI CHEN
PAULO ROBERTO MACHADO DOS SANTOS
DIEGO HIDEK CAETANO IDE

AUTOMAÇÃO DO PROCESSO DE REFINO DA MASSA DE CACAU - AMAZÔNIA 4.0

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista com ênfase em Computação.

São Paulo
2022

GUSTAVO DONNINI CHEN
PAULO ROBERTO MACHADO DOS SANTOS
DIEGO HIDEK CAETANO IDE

AUTOMAÇÃO DO PROCESSO DE REFINO DA MASSA DE CACAU - AMAZÔNIA 4.0

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista com ênfase em Computação.

Área de Concentração:
Inteligência Artificial

Orientador:
Profa. Dra. Tereza Cristina Melo de Brito Carvalho

São Paulo
2022

Catalogação-na-publicação

Ide, Diego Hidek Caetano
Automação do processo de refino da massa de cacau - Amazônia 4.0 / D. H. C. Ide, G. D. Chen, P. R. M. Santos -- São Paulo, 2022.
54 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Inteligência artificial 2.Automação 3.Deep learning I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t. III.Chen, Gustavo Donnini IV.Santos, Paulo Roberto Machado

RESUMO

O presente trabalho visa automatizar o processo de refino das massas de cacau produzidas pelo projeto Amazônia 4.0 se utilizando de visão computacional, tecnologias de processamento de vídeo e *deep learning*

Desenvolveu-se um software com rotinas de treinamento e supervisão, sendo a primeira capaz de gerar um banco de imagens através de gravações do processo e realizar o treinamento de uma rede neural construída baseando-se no modelo Xception, e a segunda responsável por supervisionar a massa, averiguando se esta está ou não pronta através da avaliação da rede neural sobre fotos capturadas em tempo real.

A rotina de treinamento é capaz de gerar um banco de imagens de aproximadamente 20.000 imagens distribuídas ao longo do processo de refino da massa de cacau, que dura por volta de 48 horas. A classificação das imagens deve ser feita manualmente. O modelo alcançou altos valores de acurácia, e seu treinamento durou menos de 2 horas.

Concluiu-se que os objetivos do trabalho foram alcançados através do software desenvolvido, capaz não somente de realizar a supervisão de massas de cacau, mas também de rapidamente treinar uma rede neural que utilize visão computacional para tomar decisões binárias.

Palavras-Chave – Amazônia 4.0. Automação. Visão Computacional. Deep Learning. ESG. Xception.

ABSTRACT

The present work aims to automate the grinding process of cocoa nibs into cocoa liquor, required by the Amazônia 4.0 project. The solution is based on computer vision, video processing technologies and *deep learning*

A software was developed with training and supervision routines. The training routine is capable of generating an image bank by processing videos taken directly from the cocoa liquor and training the neural network built based on the Xception model. The supervision routine is responsible for supervising the cocoa liquor, checking whether or not it is ready by the neural network evaluation of photos captured in real time.

The training routine is capable of generating an image bank of approximately 20,000 images distributed throughout the cocoa liquor grinding process, which lasts around 48 hours. The classification of images must be done manually. The model achieved high accuracy values, and its training happens in less than 2 hours.

It was concluded that the objectives of the work were achieved through the developed software, capable not only of supervising the cocoa liquor, but also of quickly training a neural network that uses computer vision to make binary decisions.

Keywords – Amazônia 4.0. Automation. Computer Vision. Deep Learning. ESG. Exception.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

O trabalho foi desenvolvido com o intuito de contribuir com o projeto Amazônia 4.0, que visa incentivar o desenvolvimento econômico na amazônia utilizando métodos de modo a preservar a bio-diversidade local.

Discussões sobre esse tópico já existem há alguns anos. Algumas iniciativas já foram tomadas para alcançar esse objetivo no passado, mas nenhuma delas obteve sucesso. A primeira iniciativa tomada teve foco na preservação da floresta e dos habitantes da amazônia por meio de legislações que garantiriam essa proteção. Porém, mesmo que essas novas leis protejam a floresta, essa iniciativa não promoveu a modernização nem o desenvolvimento esperados, não solucionando os problemas socio-econômicos da região.

Após essa primeira tentativa, foi criada uma segunda iniciativa para solucionar esse problema que focou no desenvolvimento local incentivando a agricultura, mineração e a produção de energia. Isso promoveu o desenvolvimento socio-econômico da região, mas foi criticado por promover desflorestamentos, roubos de terras e queimadas, se mostrando assim um método destrutivo à região.

1.1.1 Amazônia 4.0

O projeto Amazônia 4.0 visa produzir uma nova via de desenvolvimento para a Amazônia que não seja destrutiva. Seu objetivo é tomar proveito da gigantesca bio-diversidade da Amazônia e utilizar conhecimentos científicos junto de tecnologias para mobilizar a preservação da floresta e o desenvolvimento local ao mesmo tempo.

Para isso o projeto se apoia em quatro pilares, sendo eles a sustentabilidade, a ciência, a sociobiodiversidade e a indústria 4.0. Utilizando-se de estudos da floresta, da manipulação do genoma das plantas e das tradições locais, é possível explorar de forma sustentável diversos insumos que podem ser processados por fábricas no modelo indústria

4.0 seguindo técnicas e culturas tradicionais a fim de criar um produto de valor agregado com garantia de qualidade, de rastreabilidade e de sustentabilidade.

Esses produtos então poderiam ser exportados no lugar dos commodities utilizados na sua produção, arrecadando assim mais renda e gerando desenvolvimento socioeconômico local.

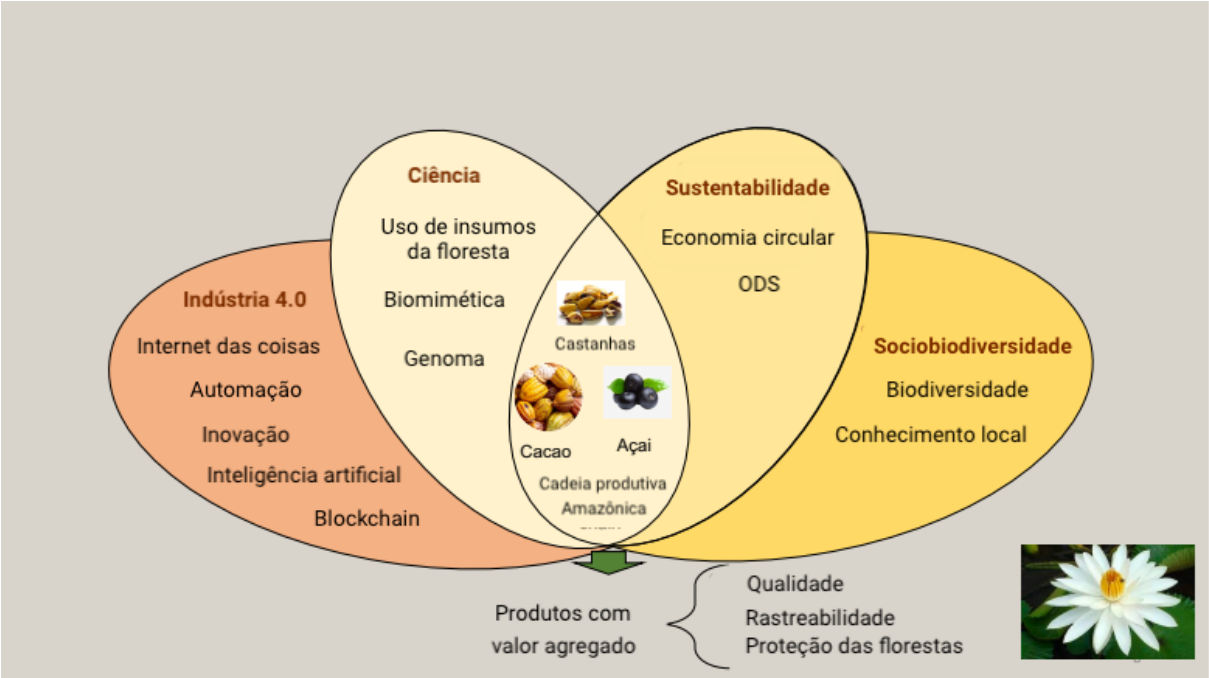

Figura 1: Pilares do projeto
Fonte: Instituto Amazônia 4.0

Esse processamento dos insumos acontecerá em laboratórios denominados Laboratórios Criativos da Amazônia (LCA) localizados na própria Amazônia. Existirão diversos LCA's focados no processamento de diversos insumos como cacau, açaí, castanha-do-pará, entre outros. Dentro desses LCA's, existirão linhas de produção condizentes com o padrão da indústria 4.0 e que possuirão interfaces para que seus usuários possam introduzir seus insumos na linha, controlar o processamento de seus insumos e recolher o resultado do processamento. Além disso, informações e parâmetros utilizados ao longo de todo o processamento dos insumos serão armazenadas em um registro local e numa blockchain.

1.1.2 Produção de chocolate

Como introduzido no tópico acima, a ideia principal do projeto Amazônia 4.0 é a agregação de valor aos insumos produzidos na floresta Amazônica, de modo a gerar desenvolvimento social e econômico para as populações locais, assim preservando a floresta

e a relação dos locais com o meio-ambiente. Sendo assim, um dos meios encontrados para alcançar tal objetivo foi a produção de chocolate. A partir do Cacau que é colhido na região e que seria vendido em seu formato natural e portanto com baixo valor agregado, podemos utilizar as fábricas para processar tal insumo e exportar um produto com maior valor agregado.

Sendo assim, nas estações do Amazônia 4.0, seria possível realizar toda a cadeia de produção do chocolate, partindo do cacau colhido por locais até o empacotamento do produto final, já dentro dos padrões de rastreamento.

1.1.3 Processo Produtivo

Para a produção do chocolate, o LCA contará com uma pequena fábrica automatizada, que cuidará de todas as etapas de produção do chocolate a partir do Cacau, desde a fermentação das sementes colhidas, até a temperagem do chocolate. Esta fábrica contará com controladores e sensores, de modo que os usuários precisem apenas fornecer as sementes do cacau que foram colhidas. Todo o processo é realizado de modo automático.

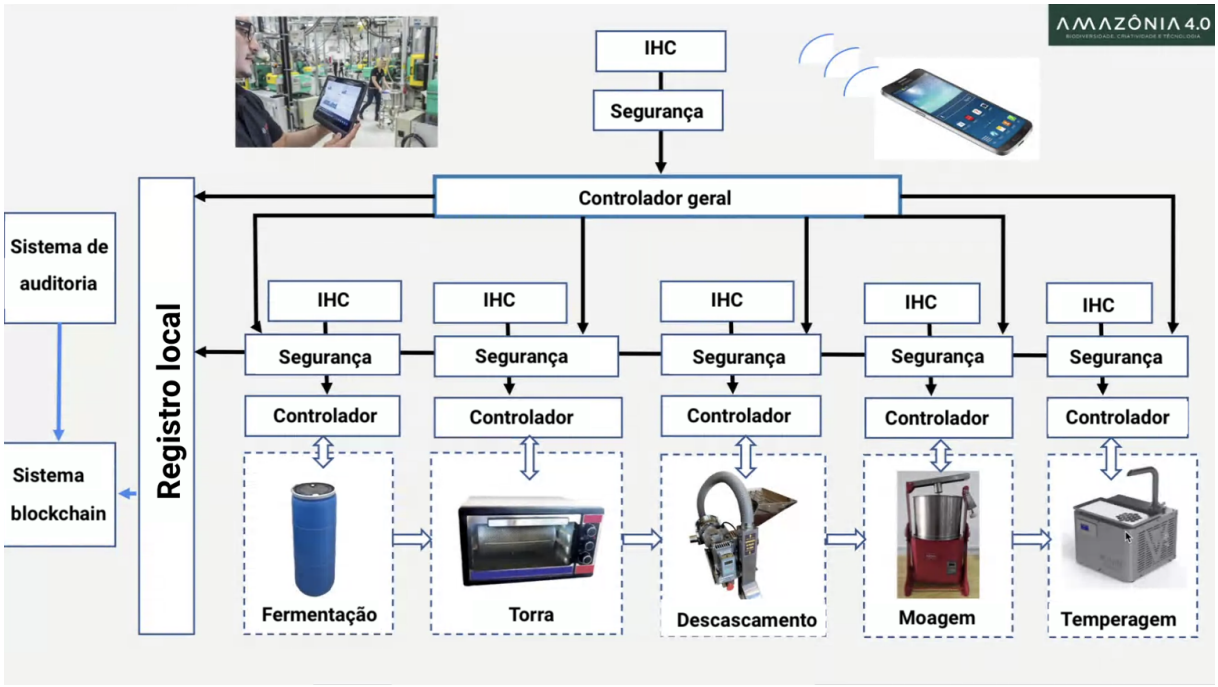

Figura 2: Processo produtivo do Cacau

Fonte: Instituto Amazônia 4.0

1.2 Objetivo

Este trabalho será responsável por automatizar o processo de moagem dos nibs de cacau presentes na linha de produção do projeto Amazônia 4.0.

Assim como mencionado anteriormente, o projeto visa instalar pequenas fábricas no padrão da indústria 4.0 dentro da região Amazônica. Por se enquadrarem no padrão 4.0, os processos e equipamentos dentro destas indústrias devem ser capazes de exercer suas atividades de forma independente através de conectividade e automação.

Uma das etapas de processamento do cacau que ainda não possui um plano claro de automação dentro do projeto é a de moagem dos nibs de cacau. Nessa etapa, os nibs (fragmentos de cacau fermentado e triturado obtido através das etapas iniciais de processamento do grão de cacau) é inserido no equipamento denominado melanger, cujo objetivo é triturar os nibs juntamente dos demais ingredientes da receita (Açúcar, leite em pó, manteiga, entre outros) afim de formar a massa base do chocolate. Essa trituração é feita através de dois cilindros e uma base rotatórios. Os ingredientes se tornam mais granulares conforme são comprimidos pelos cilindros até formarem uma massa, que é utilizada no processo de temperagem para criar o chocolate em sua forma final.

Figura 3: Interior de uma melanger.

Imagem: Clay Gordon, https://www.thechocolatelife.com

Comumente, o controle dessa etapa é feito manualmente pelo supervisor do processo

considerando a textura, brilho e tempo de moagem da massa. Nosso projeto atuará nesta etapa da produção, fornecendo meios para que essa avaliação da massa se dê de forma 100% automática e para que o processo seja interrompendo quando a massa atingir os parâmetros preestabelecidos.

1.3 Justificativa

Optamos por esse tema de pesquisa por conta de alguns fatores.

O primeiro deles é o impacto social gerado pelo projeto Amazônia 4.0. O quadro social e econômico atual da Amazônia refletem os resultados da exploração desenfreada de seus recursos naturais, pintando uma imagem de pobreza e desigualdade ao longo de seu território [1]. Acreditamos que, como estudantes da universidade com a maior classificação em pesquisa do Brasil [2], temos a responsabilidade de contribuir com pesquisas e projetos que tenham por objetivo gerar um futuro melhor para os brasileiros e para o mundo.

Outro fator que nos incentivou a pesquisar sobre esse tema é a coerência do tópico da pesquisa com os conhecimentos prévios adquiridos em nossa graduação. A solução do problema em questão, segundo nossas hipóteses atuais, funda-se fortemente em tópicos estudados no curso como processamento de imagens e inteligência artificial.

Por fim, o desenvolvimento da solução do nosso desafio pode servir como um ponto de partida para outras pesquisas no tópico de aplicação de processamento de imagens na indústria alimentícia, indústria essa que compôs 9.7% do PIB brasileiro em 2019 [3], se mostrando um tópico relevante para a economia brasileira.

1.4 Demais objetivos considerados

Antes de definir o objetivo definitivo do projeto, o grupo transitou entre alguns outros tópicos de pesquisa. Para que esse desenvolvimento não se perca, foram registrados tais tópicos nesta seção.

- Como recuperar dados de uma blockchain e criar uma interface destinada ao usuário final que apresente os parâmetros de produção de seu chocolate?
- Como prevenir leituras errôneas dos parâmetros da produção a fim de aumentar a confiabilidade dos dados da blockchain?

1.5 Organização do Trabalho

Ao longo das próximas seções, iremos introduzir os aspectos conceituais utilizados no projeto, especificar os requisitos do software desenvolvido, traçar a metodologia a ser utilizada, detalhar o desenvolvimento e por fim listar as conclusões obtidas pelo grupo.

2 ASPECTOS CONCEITUAIS

A análise em tempo real da progressão da massa do cacau apresenta grandes dificuldades técnicas. O primeiro ponto a ser abordado é em qual estágio a massa se encontra. Para determinar a fase, é necessário utilizar exemplos de imagens que representem cada estágio, criando um banco de imagens com rótulos do momento em que certas características fazem-se presentes na massa, para assim iniciar o processamento de imagem, que por sua vez irá produzir especulações da fase.

Os métodos que serão utilizados permeiam as abordagens de processamento de imagem e inteligência artificial. Processaremos as imagens registradas, a partir de modelos já produzidos pertencentes a biblioteca Keras, focada em visão computacional.

2.1 Keras

Keras é uma API de rede neural de código aberto, construída sobre o TensorFlow escrita em Python, com interface de alto nível e grande nível de profundidade, pois não é necessário a implementação de cada neurônio e utiliza camadas totalmente conectadas. Sendo assim, sendo ideal para a aplicação em nossa pesquisa.

Além de sua vasta gama de aplicações, existe grande facilidade em sua utilização por conta de sua comunidade ativa, da sua grande exemplificação de uso dos códigos e de sua capacidade de compilação em Python, permitindo a manutenção em ambientes de menor complexidade, como Jupyter Notebook e Google Colaboratory (Serviço de nuvem para aprendizado de máquina e inteligência artificial), possibilita que mesmo após o desenvolvimento do projeto pela equipe, a continuidade e ampliação da aplicação possa ser realizada por outros colaboradores do projeto.

2.2 Rede neural

Redes neurais são uma aplicação de inteligência artificial onde um computador aprende a processar e extrair informações de dados. São compostas de nós, também chamados de neurônios, e ligações que os conectam e determinam as regras de interação entre as camadas. Juntas, as camadas atuam como um sistema adaptativo para aprender com os erros e se aprimorar ao longo do tempo, sendo capazes de solucionar problemas mais complexos como identificação de imagens.

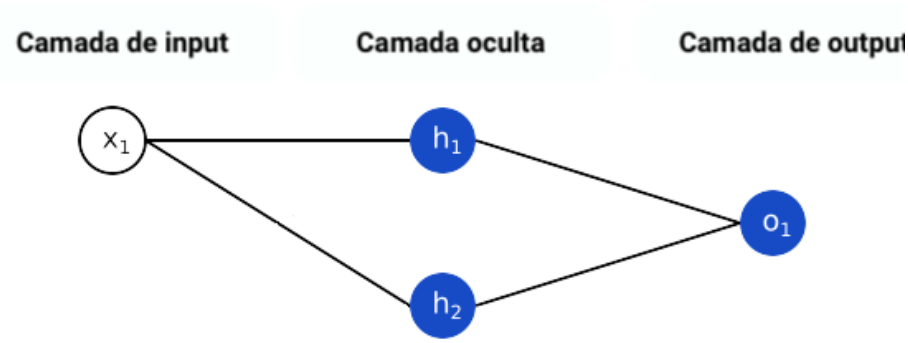

Figura 4: Rede neural simples

Imagem: ActiveState, https://www.activestate.com/resources/quick-reads/what-is-a-keras-model/

A partir do estímulo inicial, diversas camadas processam em um sistema ponderado com regras a fim de de atingir o resultado almejado. Para isso é feito um treinamento com diversas imagens corretas para que a lógica e os pesos sejam ajustados ao longo do treinamento.

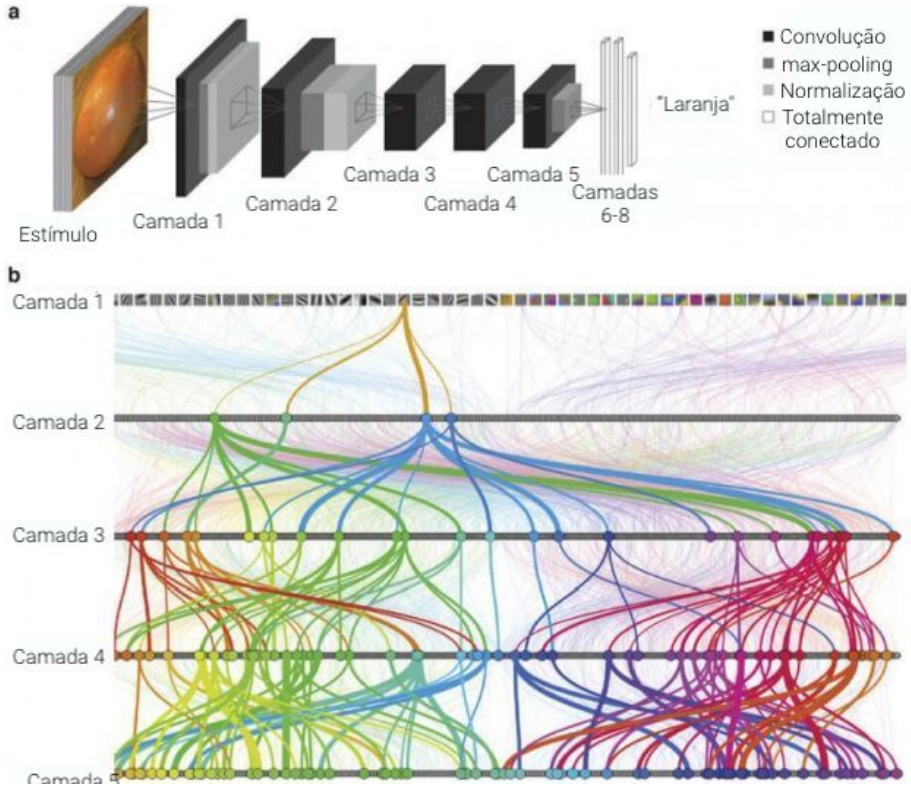

Figura 5: Camadas da rede neural

Imagem: ActiveState, https://www.activestate.com/resources/quick-reads/what-is-a-keras-model/

2.3 CNN

CNN é a abreviação de *Convolutional neural network*, um algoritmo de aprendizado profundo aplicado a visão computacional que extrai as informações da entrada (*Inputs*), separa em camadas e as junta de novo após os processamentos. Essa prática para classificação de imagens tem desempenho melhor do que uma rede neural de alimentação direta totalmente conectada. O algoritmo de rede neural totalmente conectado tem todos seu nós de uma camada conectados com os nós da camada seguinte, sem assumir suposições, enquanto a rede convolucional assume que a entrada será uma imagem e atribuí pesos aos neurônios e classifica sua importância.

2.3.1 Convolução e Convolução separável

A convolução é o processo de transformar uma imagem aplicando um *kernel* sobre cada ponto e arredores, sendo o *kernel* uma matriz de valores cujo tamanho e valores determinam o efeito de transformação do processo de convolução. Essa transformação é análoga a uma aplicação de filtro em cada região da imagem, assim produzindo uma

nova imagem com características que reforcem determinado aspecto dependendo do filtro durante o processo.

A convolução separável é composta por duas convoluções, uma em profundidade e uma ponto a ponto. A convolução em profundidade processa a imagem e resulta em um novo tensor de mesma dimensão que o tensor de entrada (neste projeto, resulta em um tensor (180, 180, 3)). Cada *kernel* é iterado sobre um canal da imagem de entrada, analisando as relações entre canais.

A convolução ponto a ponto, por sua vez, é aplicada sobre todas os canais do tensor de entrada em apenas uma única posição, resultando em um novo tensor de um único canal (neste projeto, resulta em um tensor (180, 180, 1)). O *kernel* é iterado sobre toda a região da imagem, analisando suas relações espaciais.

Comparada à convolução convencional, a convolução separável apresenta a vantagem de ser capaz de processar as relações espaciais e entre canais de forma desacoplada [4], economizando operações computacionais e, consequentemente, tempo. A convolução separável modifica a imagem apenas uma vez ao longo da convolução de profundidade, e depois alonga a quantidade de canais da saída através das convoluções ponto a ponto. A convolução convencional normal transforma a imagem diversas vezes para atingir esse objetivo.

2.3.2 *Pooling*

A partir da camada de convolução, recebe suas características e produz um mapa condensado. Esse processo seleciona os elementos mais impor antes da imagem a partir dos atributos recebidos pelo processo anterior, como um resumo da camada,para que assim, possam ser unidas no processo final. A vantagem de utilizar essa metodologia é de que elementos ruidosos dentro da imagem não interferem ou interferem pouco no resultado final, pois são colocados em pesos menores dentro do processamento.

2.3.3 *Batch normalization*

O objetivo de utilizar essa normalização é a aceleração da convergência das informações, atualmente não se sabe exatamente o motivo desta propriedade, a melhor hipótese é de que torna o cenário de otimização mais suave, tornando o treinamento de descida de gradiente mais eficiente. Isso, por sua vez, permite o treino de uso de taxas de aprendizado mais altas do que sem a normalização. Em linhas gerais, o algoritmo faz uma medição de

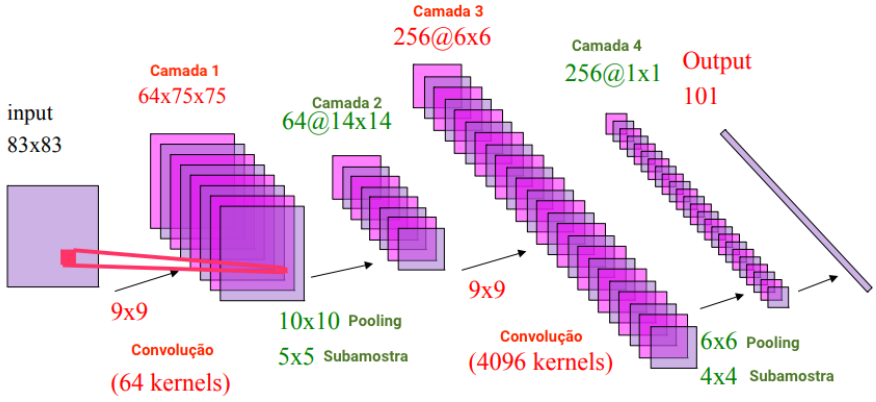

Figura 6: Exemplo de Convolução e Pooling
Imagem: Marcos Tanaka,
https://imasters.com.br/back-end/classificacao-de-imagens-com-deep-learning-e-tensorflow

variância e descarta possibilidade muito distantes da almejada.

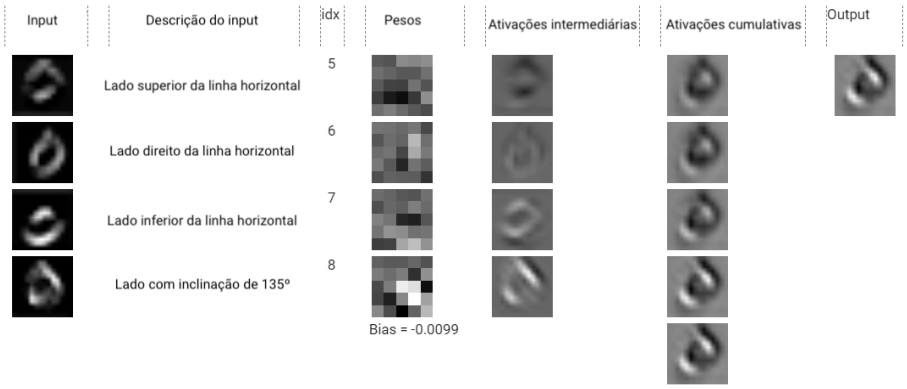

Figura 7: Exemplo de Rede Neural Convolucional
Imagem: Nazar Ilamanov,
https://towardsdatascience.com/explainable-mnist-classification-dissection-of-a-convnet-f32910d52842

2.3.4 Camada ReLU

A camada ReLU é a abreviação para Unidade Linear Retificada, é uma etapa complementar aos tópicos acima. O retificador serve para separar a linearidade, uma vez que as imagens em seu contexto não são lineares, em seu uso aplica a função não saturante, removendo valores negativos e auxiliando que o uso computacional não aumente exponencialmente a operação de uma rede neural, sem proporcionar danos significativos na precisão e possibilitando uma operação muito mais rápida.

2.4 Arquitetura *Xception*

Essa arquitetura é baseada na profundidade das camadas de convolução separadas, inspirada da arquitetura *Inception V3* [5]. A *Inception V3* é uma das arquiteturas de redes neurais mais famosas, por ser planejada em torno do banco de dados *ImageNet*, banco de dados público que contem mais de 20 mil categorias e mais de 19 milhões de fotos. Entretanto a *Xception* tem uma performance um pouco superior a da *Inception V3* utilizando o banco de imagens da *ImageNet* e razoavelmente maior em outros bancos de imagens, essa superioridade de performance se deve a um uso mais eficiente dos parâmetros do modelo.

Ambas arquiteturas tem como hipótese um único *kernel* de convolução ser encarregado de mapear simultaneamente as correlações entre canais e as correções espaciais, uma vez que uma camada de convolução tenta aprender filtros em espaço 3D com duas dimensões espaciais(altura e largura) e uma dimensão de canal. A finalidade do módulo *Inception* é particionar explicitamente esse processo em uma série de operações que examinariam separadamente as correlações entre canais e as correlações espaciais para torná-lo mais simples e eficaz. Mais especificamente, o módulo *Inception* típico examina primeiro as correlações entre canais usando um conjunto de convoluções 1x1, particionando os dados de entrada em três ou quatro espaços 3D menores do que o espaço de entrada original e, em seguida, mapeia todas as correlações nesses espaços 3D menores usando convoluções 3x3 ou 5x5 regulares, em suma, as correlações entre canais e as correlações espaciais são suficientemente desacopladas que é preferível não mapeá-las em conjunto. Esse modelo é dividido em três partes ou *flows*, entrada, intermediária e saída, como demonstrado na figura 8.

A camada de entrada recebe a imagem que será separada nas faixas do vermelho, verde e azul. Após a primeira separação e convolução, onde é aplicado um filtro 2x2, são realizadas operações de separação das camadas, transformação ReLU, passando por uma nova convolução e transformação ReLU. Em uma nova etapa, duas operações são realizadas em paralelo, a convolução espacial e duas convoluções separadas, com linearização entre elas, e um *MaxPooling* (forma onde são utilizados os resultados mais significativos para a próxima etapa, as duas operações em paralelo são unidas para um processo parecido, diferenciando pela linearização antes das convoluções separadas. Esse processo é realizado mais uma vez até o momento final dessa primeira etapa onde a imagem agora apresenta largura e altura menores do que a imagem inicial, mas possui 728 dimensões.

A camada intermediária a imagem já se encontra altamente seccionada e são realizados

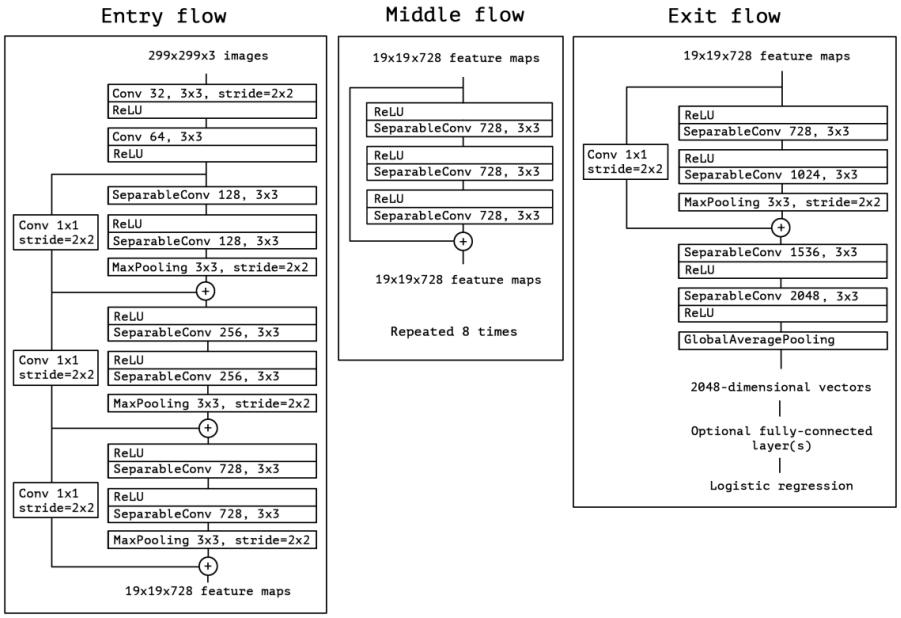

Figura 8: Xception Flows
Imagem: François Chollet, https://arxiv.org/pdf/1610.02357v3.pdf

as transformações ReLU, convoluções separadas três vezes, antes de unir com a convolução espacial, sem uso do sendo *MaxPooling*, repetido 8 vezes. Essa etapa tem como objetivo destacar elementos da imagem que possam ser importantes e que sejam realizados com maior eficiência do que no bloco anterior.

A camada de saída é a fase de preparação do resultado onde já aplicado a maior parte da rede neural, é realizado um processo parecido com a última parte da camada de entrada e por fim, realiza mais duas convoluções separadas seguidas de linearizações com um *GlobalAveragePooling*, para sintetizar a média de cada canal em uma camada de neurônios totalmente conectada formando os pesos das características definitivas para que a saída identifique finalmente quais sãos as características definitivas para tomada de decisão.

2.5 *Grad-CAM*

Grand-CAM é um método de computação visual, seu nome se refere à *Gradient class activation mapping*, e primeiramente foi utilizado para entender a razão de classificação de imagens. Esse método utiliza a imagem de entrada e cria um modelo, onde é cortado na camada onde se deseja aplicar esse método, anexa as camadas totalmente conectadas para previsão, e ao final adiciona essas camadas na saída de forma que fiquem sobrepostas no resultado final. Assim, podemos visualizar de forma intuitiva quais regiões foram mais

utilizadas para analisar determinadas imagens, um exemplo disso é uma imagem com um cachorro e outros elementos, quando aplicado esse método é possível visualizar um grande uso de partes da imagem focadas no animal em si e em particularidades desse animal, como seu focinho e seu pelo. Esse método permite a depuração da imagem para entender o viés utilizado pelos neurônios e seus pesos, dentro do projetos facilita o reconhecimento da abordagem utilizada e a confirmação de utilização dos parâmetros pretendidos.

Figura 9: Mapa de calor de cachorro

Imagem: glassboxmedicine,

https://glassboxmedicine.com/2019/06/11/cnn-heat-maps-class-activation-mapping-cam//

2.6 Bash

Bash, ou *Bourne-Again Shell*, é a *shell* padrão de grande parte das distribuições linux e é o programa responsável por executar os comandos feitos pelo usuário em sua linha de comando. Scripts escritos em bash consistem em sequências de comandos que devem ser executados linha a linha pelo terminal.

Apesar de não oferecer funcionalidades de alto nível, a linguagem bash permite a utilização de variáveis, manipulação de strings, loops e diversas outras funcionalidades básicas presentes em linguagens de programação.

2.7 FFmpeg

FFmpeg é um software *open source* que consiste em uma suíte de bibliotecas extremamente versátil para manipular vídeos, áudios e outros arquivos multimídia, suportando praticamente qualquer formato de conteúdo multimídia [6]. O projeto FFmpeg oferece diversas ferramentas completas e interfaces de linha de comando (CLIs) para tratar diferentes tipos de arquivo.

Neste projeto, foi utilizada a ferramenta de linha de comando *ffmpeg*, responsável por dissecar vídeos e extrair todos os seus frames um formato .jpg.

2.8 Regex

Expressões regulares, *Regular Expressions*, *regex* ou *regexp* são sequências de caracteres que especificam um padrão de busca em um determinado texto. Esse tipo de expressão é normalmente utilizado para manipular textos automaticamente, encontrando determinados elementos e os substituindo por outros dentro de textos grandes.

3 METODOLOGIA

A fim de guiar o desenvolvimento do trabalho, o grupo definiu uma metodologia a ser seguida. Inicialmente, seria realizado um estudo sobre como os supervisores de linhas de produção de chocolate artesanal realizam a avaliação manual de suas massas de cacau, a fim de levantar os requisitos funcionais do *hardware* e definir as tecnologias a serem utilizadas na solução.

Para realizar esse estudo, o grupo agendaria entrevistas com chocolateiros e chocolateiras tanto integrantes do projeto Amazônia 4.0 quanto externos ao projeto. Através dessas entrevistas, devem ser levantadas as características mais relevantes para analisar uma massa de cacau, e a partir dessas características, o grupo seria capaz de definir o *hardware* necessário com maior propriedade.

Após obter o *hardware* necessário, o grupo buscaria definir a arquitetura da rede neural a ser utilizado e, paralelamente a isso, adquirir um banco de imagens da massa de cacau ao longo de uma moagem completa (do momento que a máquina foi ligada até o momento em que for desligada) através de gravações de autoria própria própria melanger onde a automação será aplicada. A partir dessas gravações, seria possível decodificar o vídeo e extrair os *frames* que o compõe, gerando assim o banco de imagens.

Possuindo o banco de imagens, seria necessário identificar a imagem onde o processo de refinamento da massa se encerra para que o banco de imagens possa ser dividido entre as imagens da massa em refinamento e as imagens da massa pronta.

Feita a divisão, o grupo desenvolveria e treinaria uma rede neural que deve ser capaz de inferir se uma massa de cacau retratada em uma foto está ou não pronta através de uma classificação binária. Utilizando o banco de imagens, o algoritmo deverá ser capaz de identificar padrões característicos de massas de cacau prontas e não prontas, e utilizá-los para realizar a inferência.

4 ESPECIFICAÇÃO DE REQUISITOS

O sistema que foi desenvolvido deve trabalhar de forma autônoma e independente, sendo capaz de identificar o estados da massa de cacau utilizada na produção de chocolate a partir da análise de imagens em tempo real.

4.1 Funcionais

O sistema terá como objetivo controlar o processo de refino da massa de cacau produzida através de análise de imagens. Tendo isso em mente, foram levantados os seguintes requisitos funcionais:

- Um usuário deve ser capaz de treinar e retreinar o modelo de rede neural através de sua interface.
- O software deve conseguir capturar vídeos e imagens através de câmeras controladas por bibliotecas de visão computacional.
- O software deve ser capaz de manipular imagens quando necessário, aplicando transformações sobre lotes de imagens automaticamente.
- Utilizando-se das imagens, o software deve ser capaz de treinar automaticamente um modelo de rede neural para classificar a massa entre "pronta"e "não pronta"em tempo hábil para que não hajam intervalos grandes de tempo entre a conclusão do refino da massa e a resposta do sistema.
- O software deve oferecer meios de analisar o desempenho da rede neural.
- O usuário também deve poder executar o software em modo de supervisão através de sua interface.
- As imagens adquiridas através da câmera devem ser alimentadas à rede neural automaticamente, averiguando se a massa está ou não pronta.

- Ao identificar que a massa está pronta, o software deve mandar um sinal à melanger, encerrando assim o processo de refino.

4.2 Usabilidade

Devido a característica autônoma do projeto, nosso sistema atuará de forma automática em uma etapa específica do processo de produção, não necessitando portanto de interações diretas dos usuários com os equipamentos nem com o software ao longo do processo. Sendo assim, a única especificação de usabilidade necessária é expor uma interface do programa na qual o usuário possa controlar o funcionamento do software, alterando seus parâmetros.

4.3 Confiabilidade

Como característica mais fundamental do projeto, o sistema precisará apresentar alto nível de confiabilidade, apresentando baixa frequência de falhas e alta precisão em seu funcionamento. Deste modo o sistema deve ser robusto e preciso, possuindo um baixo nível de erros.

4.4 Manutenibilidade

Devido às condições de uso, os equipamentos se encontrarão em regiões afastadas e de difícil acesso, requisitando portanto o uma baixa frequência de manutenção. Problemas de software devem ser solucionáveis através programas de acesso remoto.

5 DESENVOLVIMENTO

O grupo considera que o desenvolvimento do projeto iniciou-se antes mesmo da definição do tema abordado neste trabalho, tendo início no momento em que o grupo começou a definir o escopo do trabalho dentro do projeto Amazônia 4.0. Por conta da liberdade de tema que o Amazônia 4.0 oferece, essa tarefa foi atacada pelo grupo através de conversas com sua orientadora, com outros integrantes do Amazônia 4.0 e com professores de disciplinas que tangiam os objetivos considerados.

Após a consideração de dois outros temas de trabalho, foi definido pelo grupo em conjunto da orientadora o objetivo da equipe e iniciado o desenvolvimento da solução do tema definido.

5.1 Levantamento de hipóteses

O grupo iniciou suas atividades levantando hipóteses sobre como a rede neural deveria atuar no projeto. Duas hipóteses foram inicialmente consideradas:

1. Através de imagens microscópicas da massa de cacau, a rede deve conseguir identificar continuamente a granularidade dos nibs presentes na massa através de seu diâmetro. A partir da média dos diâmetros identificados nas imagens, a rede deve definir se o processo deve ser continuado ou encerrado.

2. Através de imagens da massa de cacau, consiga classificar continuamente a textura, o brilho e a homogeneidade da massa. Usando esses parâmetros junto do tempo decorrido desde o início da moagem, a rede deve definir se o processo deve ser continuado ou encerrado.

A primeira hipótese apresentou grande potencial de gerar uma solução genérica para o problema, facilitando sua aplicação em outros pontos da produção ou até mesmo em outras linhas de produção onde esse controle de granularidade é desejado. Porém, acompanhada

de sua generalidade, observa-se uma maior complexidade adicionada ao projeto. Como os nibs são moídos juntos de outros ingredientes, surge a necessidade de implementar métodos que sejam capazes de diferenciar nibs dos demais compostos da massa. Além disso, os nibs alcançam diâmetros extremamente pequenos ao longo da moagem, diâmetros esses muito menores do que uma câmera usual é capaz de captar. Por esse motivo, a utilização de equipamentos mais específicos se tornaria mandatória.

A segunda hipótese não apresentou tanta especificidade em seus pré-requisitos, mas também gerou uma solução genérica para o problema.

Para definir como a rede neural seria aplicada, o grupo recorreu a opinião de terceiros através de entrevistas.

5.2 Entrevistas e visitas técnicas

Ao longo do projeto, procuramos realizar entrevistas com diferentes chocolateiros e chocolateiras afim de entender e aprender mais sobre o processo da produção do chocolate de modo a implementarmos a melhor decisão.

Inicialmente, conversamos com um produtor de chocolate do interior de São Paulo e integrante do projeto Amazônia 4.0, que nos apresentou aos conceitos de ponto de refino e ponto de conchagem do chocolate, dividindo o período no qual o chocolate fica na *melanger* em 2 diferentes momentos. No primeiro momento, todos os ingredientes são adicionados, triturados e misturados, e são analisados pelo chocolatier visualmente através de aspectos como seu brilho, textura e viscosidade, em um processo que pode demorar tempos distintos a depender das condições e características dos ingredientes, alcançando durações entre 24 a 48 horas. Na etapa seguinte, denominada conchagem, o chocolate permanece dentro da *melanger* por um tempo fixo determinado através do volume de ingredientes, sendo portanto uma etapa que não exige grandes implementações por parte de nossa solução.

Conversamos também com um representante da empresa NEC, uma das maiores provedoras globais de soluções integradas de tecnologia da informação e comunicação [7], com o objetivo de discutir sobre as hipóteses de como a rede neural deveria ser aplicada ao projeto, assim como o *hardware* que deveria ser utilizado. Através da conversa, percebeu-se que a rede neural não precisa medir parâmetros da massa para realizar suas inferências, podendo simplesmente ser uma rede neural convolucional capaz de identificar diretamente através de fotos se o refino deve ou não ser continuado, definindo assim como a rede neural

seria aplicada ao projeto.

Além disso, baseando-se nas especificações do projeto, concluímos que a utilização de uma *webcam* em conjunto com uma fonte de luz dentro da *melanger* apresentariam muito provavelmente um desempenho satisfatório. Caso o desempenho da *webcam* não fosse satisfatório em produção, foi definido também nesta reunião que uma câmera tipo CCTV deveria ser utilizada.

Essas conclusões foram extremamente importante pois indicaram que nenhum equipamento de difícil aquisição deveria ser comprado e especificaram a abordagem do grupo à solução, permitindo que o grupo planejasse o início do desenvolvimento dos scripts.

O grupo também tentou entrar em contato com 3 lojas produtoras de chocolate no estado de São Paulo na tentativa de agendar visitas técnicas às suas cozinhas e aprender mais sobre as diferentes variações das técnicas de produção. Infelizmente, nenhuma das lojas demonstrou interesse em ajudar o grupo. Acreditamos que isso tenha acontecido por conta da sensibilidade das informações que o grupo buscava, afinal, a receita do chocolate de cada uma das lojas é única e não é aberta ao público.

Por fim, tentamos agendar uma visita técnica à fábrica da empresa Omega7, responsável pela produção dos equipamentos que serão utilizados no projeto Amazônia 4.0, para estudar a estrutura da *melanger*, definir onde a câmera seria inserida no equipamento e fazer barras de chocolate de teste, passando por um processo de refino inteiro. O grupo cobiçou muito a leva teste de chocolate, pois seria uma ótima oportunidade para utilizar suas próprias rotinas de software para gravar o processo de refino, gerar um banco de imagens completo e realizar um treinamento do modelo com imagens reais, gerando assim uma análise mais rica dos resultados do modelo. Porém, diversos fatores dificultaram tal visita técnica.

O primeiro deles foi a distância até a fábrica, localizada em São José dos Campos. Como os integrantes do grupo são residentes de São Paulo, a ida até a fábrica não foi algo trivial. O segundo foi a disponibilidade do chocolateiro responsável por supervisionar a leva teste de chocolate, que também não é morador de São José dos Campos e não dispôs de nenhum horário livre para produzir a leva. O último fator foi a duração do processo, que demora até 48 horas para terminar e obriga o grupo a ir e voltar da fábrica duas vezes.

Considerando todas essas dificuldades, o grupo rediscutiu a metodologia planejada e concluiu que o banco de imagens deveria ser gerado de forma alternativa para que o projeto pudesse ser concluído dentro do cronograma. Definiu-se que o banco de imagem seria

inicialmente composto por imagens retiradas de vídeos de refino de chocolate encontrados na internet, e que seria implementada uma rotina que permita o retreinamento do modelo com imagens reais adquiridas na linha de produção (chamada de "rotina de treinamento").

5.3 Aquisição dos vídeos

Para criar tanto a rotina de treinamento quanto a que supervisiona o estado da massa (chamada de "rotina de supervisão"), seria necessário desenvolver um método de captura de imagens da massa de cacau dentro da *melanger*. O grupo optou por iniciar o trabalho por este ponto pois as imagens da massa serão necessárias para o desenvolvimento de todas as etapas seguintes do projeto. Assim, foram discutidos os métodos que deveriam ser adotados para a aquisição de imagens da massa de cacau em duas situações distintas.

A primeira delas é na gravação do processo de refinamento completo, a fim de obter um banco de imagens. Por se tratar de um processo que demora até 48 horas, uma gravação contínua em alta resolução geraria vídeos que ocupariam muita memória, se mostrando um método inviável. Para se aproveitar da extensão do processo, o grupo planejou executar a aquisição de imagens através de pequenas gravações distribuídas ao longo do processo de refino inteiro, e assim se definiu que seria feito um programa que, através de uma *webcam* conectada ao dispositivo que o rode, faça gravações de 5 segundos a cada 10 minutos, até o fim do processo. Feita a gravação, seria feito outro programa capaz de extrair automaticamente os *frames* das diversas gravações.

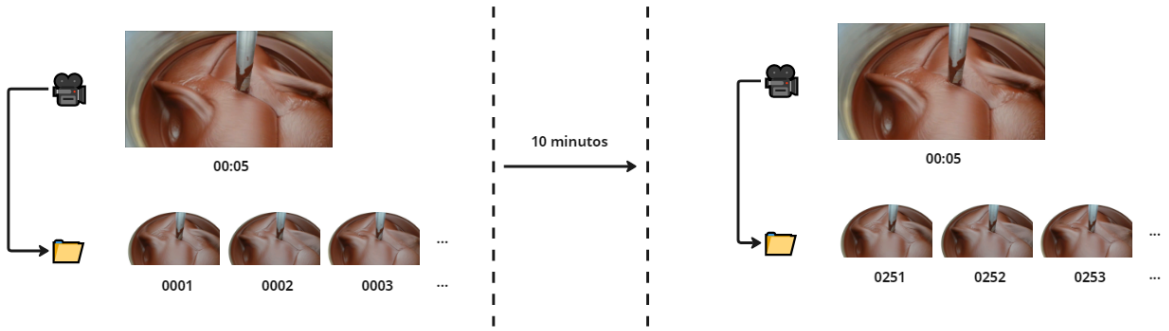

Figura 10: Ilustração do processo de obtenção do banco de imagens.

Por conta da biblioteca utilizada (cv2) algumas dificuldades foram encontradas na criação desta funcionalidade, sendo a principal delas no registro de cada um dos *frames* capturados. Após os primeiros testes, percebemos que os vídeos resultantes das gravações estavam acelerados, e a duração total de cada um dos vídeos não correspondia à duração

especificada ao programa. Através de pesquisas, foi descoberto que a função capaz de agrupar os *frames* em um vídeo oferecida pela biblioteca é pouco eficiente e seu tempo de execução acaba afetando a quantidade de *frames* por segundo que a *webcam* captura, consequentemente desalinhando a quantidade de *frames* capturados pela web e o valor de *frames* por segundo definida no vídeo finel, gerando a aceleração do vídeo. Porém, como os vídeos serão processados nas etapas seguintes do trabalho, essa aceleração não foi um problema. O problema foi contornado determinando-se experimentalmente a duração de gravação definida ao script que resultasse em um vídeo final de 5 segundos.

A segunda situação considerada é o software, já em produção, inferindo se o refinamento já foi concluído. Nesta situação, por necessitar apenas algumas poucas amostras da massa, uma simples foto seria suficiente para realizar a inferência. Porém, como ainda veremos adiante, optou-se por extrair mais do que uma amostra a fim de se obter máxima precisão na inferência. Sendo assim, o grupo desenvolveu um programa que realiza a captura de múltiplas fotos da massa ao longo de 10 minutos, realiza a inferência sobre cada uma das fotos e forneça uma inferência final baseada no resultado da maioria das inferências individuais.

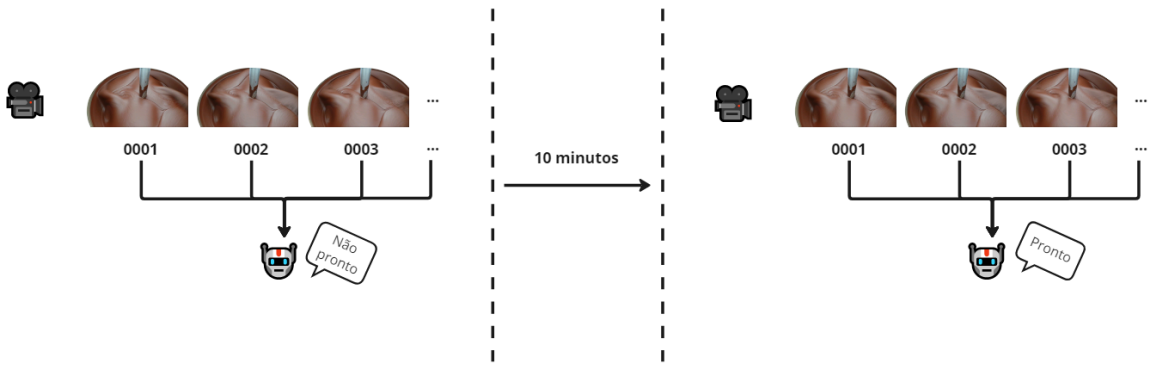

Figura 11: Ilustração do processo de obtenção de imagens para averiguação.

Esta etapa do desenvolvimento gerou a estrutura inicial da solução, que será construída progressivamente ao longo dos próximos tópicos desta seção através do diagrama a seguir:

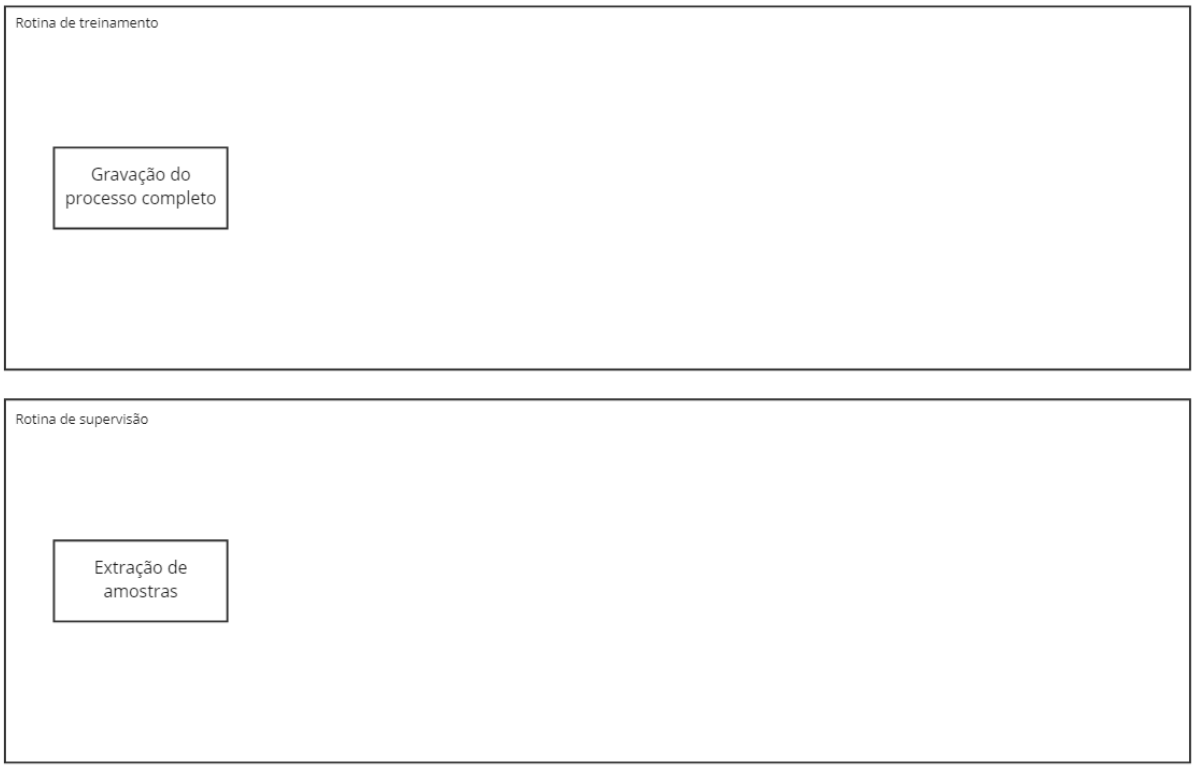

Figura 12: Estrutura da solução após desenvolvimento de scripts de aquisição de vídeos

5.4 Aquisição de imagens

Assim como discutido anteriormente, a aquisição de imagens do processo de refino da massa de cacau através de gravações *in loco* seria responsável por fornecer informações chave para o treinamento do modelo, mas por se tratar de uma gravação que não depende apenas do grupo, se perderia muito tempo combinando essa tarefa. Por esse motivo, o grupo julgou apropriado realizar os treinamentos iniciais do modelo usando imagens extraídas de vídeos encontrados na internet e que retratam o processo de refinamento da massa de cacau por inteiro, normalmente em formato de *timelapse*.

A utilização desses vídeos foi extremamente vantajosa pois agilizou o início do desenvolvimento do código do modelo e dos códigos dos programas auxiliares que foram utilizados para extrair *frames* de vídeos automaticamente e para aplicar máscaras de corte sobre tais *frames*. Além disso, esse adiantamento nos forneceu mais imagens tratadas para alimentar o modelo caso seu desempenho utilizando apenas as imagens retiradas da gravação feita pelo grupo não seja satisfatório, fornecendo uma maior diversidade de qualidades de gravação, ângulos de filmagem, iluminação e até mesmo formatos de equipamento.

Para realizar a extração dos *frames*, o grupo criou um script na linguagem *bash* que se utiliza de uma ferramenta multimídia chamada **FFmpeg** para realizar a decodificação dos vídeos, e de lógicas implementadas diretamente eh *bash* para definir os parâmetros de decodificação dos vídeos. Optamos por escrever esse script em *bash* pois a ferramenta FFmpeg é uma ferramente CLI, ou seja, que expõe suas funcionalidades através da linha de comando do terminal, e a melhor forma que o grupo encontrou para automatizar esse processo foi justamente utilizando uma linguagem que tenha acesso direto à linha de comando.

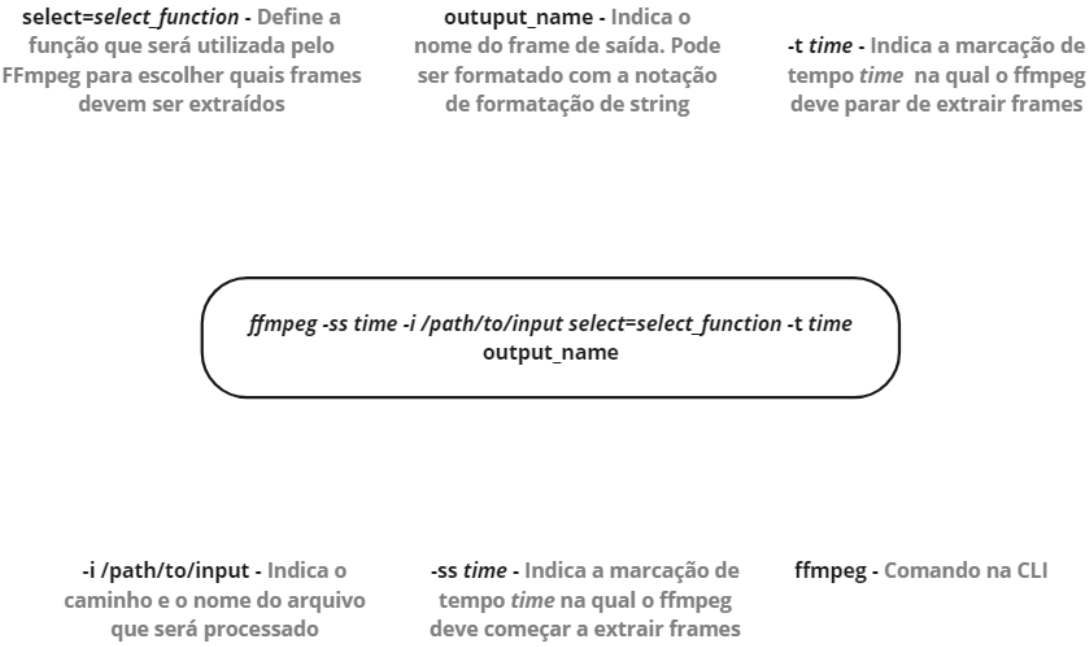

Figura 13: Diagrama explicativo da CLI do FFmpeg.

O script de extração de *frames* de vídeos da internet se baseia em uma estrutura de pastas específica para fazer o processamento correto dos vídeos. Essa estrutura pode ser visualizada no diagrama a seguir.

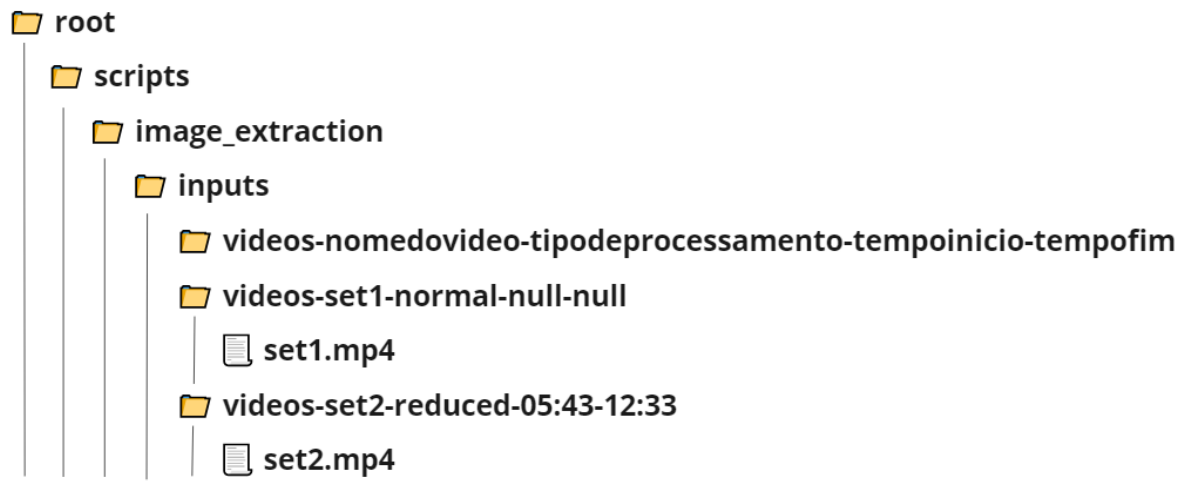

Figura 14: Estrutura de pastas do script.

Observa-se uma pasta chamada "image_extraction", e dentro desta temos as pastas de input de vídeos chamada "inputs"e a pasta onde o resultado do processamento será armazenado chamada "outputs". Dentro da pasta de inputs, temos pastas com uma nomenclatura seguindo o padrão especificado na imagem, ou seja, pastas cujos nomes são compostos por "videos-"escrito por extenso, pelo nome do arquivo de vídeo (.mp4) que deverá ser processado, pelo tipo de processamento, pela marcação de tempo onde o FFmpeg deve iniciar o processamento do vídeo e, por fim, pela marcação de tempo onde o processamento deve se encerrar. Dentro de cada uma dessas pastas deve existir o arquivo de vídeo que será alvo do processamento, e esse arquivo deve possuir o mesmo nome que o nome especificado no título da pasta.

Dentro da pasta de outputs, temos pastas cujos nomes são compostos por "images-"escrito por extenso e pelo nome do arquivo de vídeo que foi processado. Dentro de cada uma dessas pastas, teremos todos os frames extraídos do input cujo nome é igual ao nome da pasta.

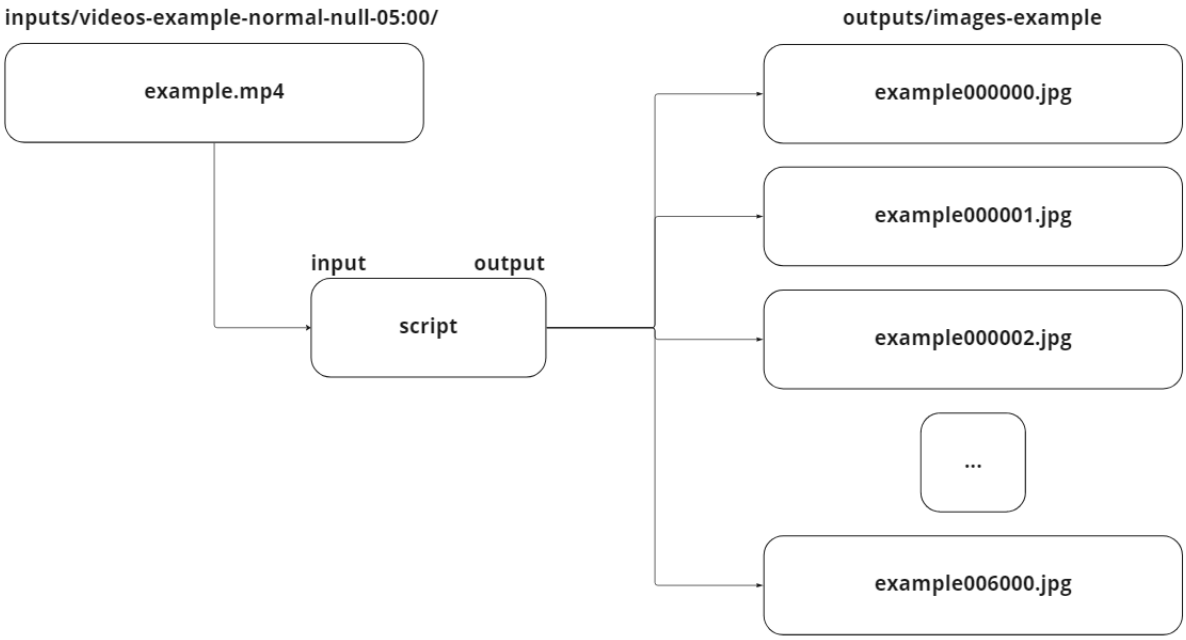

Figura 15: Funcionamento do script.

A estrutura e os nomes das pastas são de extrema importância pois o script lê os parâmetros de processamento dos vídeos diretamente desses nomes. Quando uma pasta é nomeada "videos-example-normal-null-05:00", por exemplo, uma expressão regular (Regex) é utilizada para analisar a frase e separar seus componentes. Dentro do script, os valores "example", "normal", "null"e "05:00"são atribuídos a variáveis, que por sua vez são utilizadas para compor o comando que executará o processamento. Valores "null"significam que o parâmetro não foi definido pelo usuário, e não será considerado na hora do processamento.

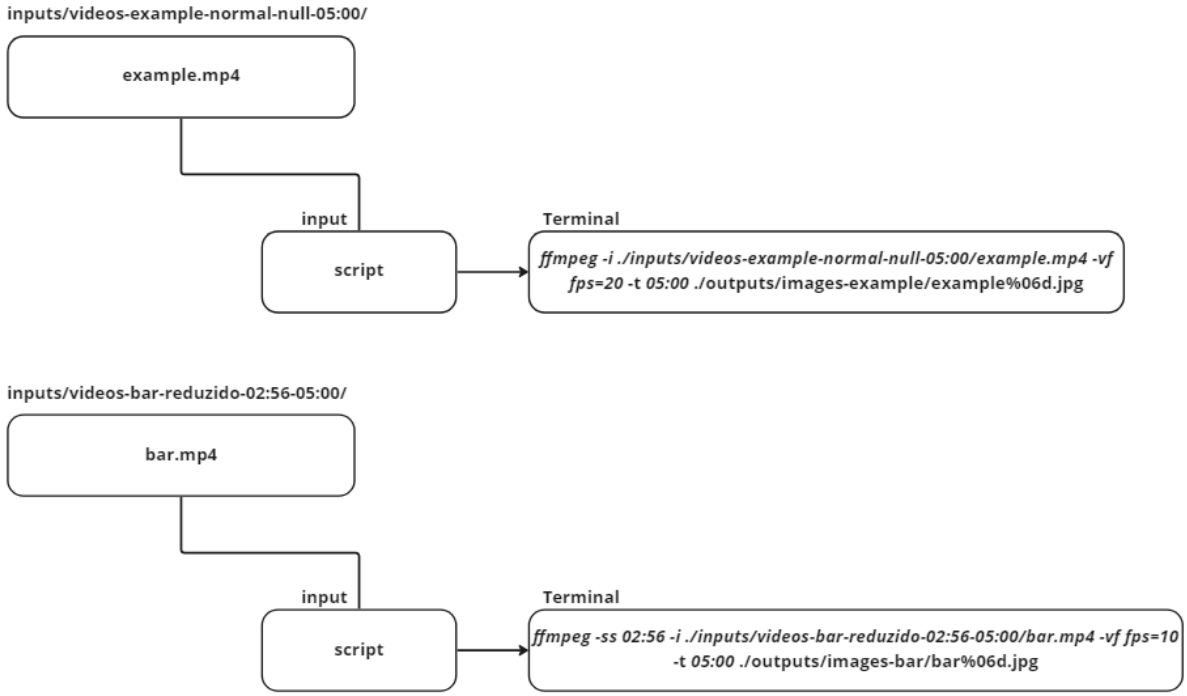

Figura 16: Exemplo de comando gerado pelo script

Utilizando o script, o grupo foi capaz de extrair 1696 imagens a partir de um vídeo *timelapse* encontrado na internet [8]. Com base nesse valor, é possível afirmar que ao extrair as imagens das gravações resultantes do script de aquisição de vídeos, serão geradas aproximadamente 20.000 imagens para o banco de imagens definitivo.

Esta etapa do desenvolvimento complementou o diagrama da solução, adicionando a ele mais uma etapa.

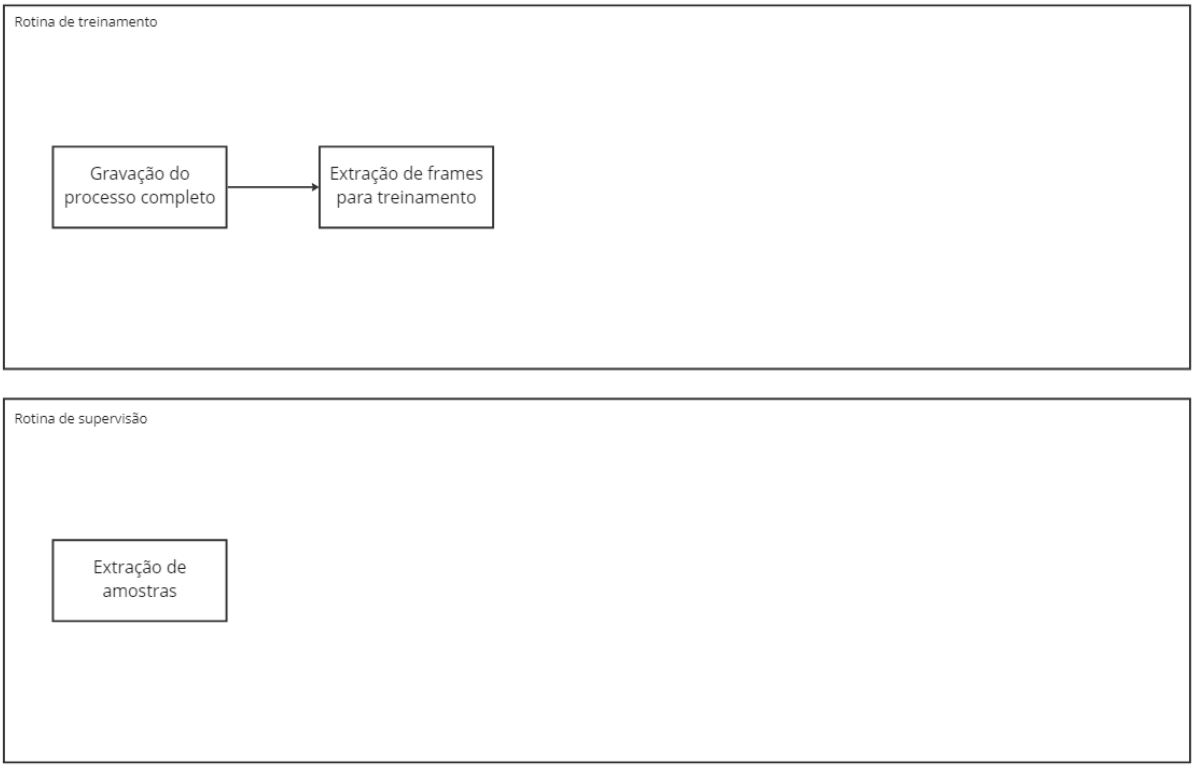

Figura 17: Estrutura da solução após desenvolvimento de script de extração de imagens

5.5 Pré-processamento das imagens

Uma etapa muito importante para a criação de um bom viés na rede neural é a de pré-processamento das imagens. Optamos por isolar a região de interesse das imagens e excluir as regiões sem informação relevante, ou seja, excluir todas (ou quase todas) as regiões das imagens que não ilustram a massa de cacau. Para isso, o grupo criou um script que aplica máscaras em todas as imagens em uma determinada pasta e salva os resultados em uma pasta de output.

A máscara consiste em uma imagem criada em um programa de edição de imagens, elaborada pelo próprio grupo, de resolução semelhante aos frames extraídos dos vídeos e composta por uma região branca que será aplicada à todas as imagens da pasta de input como um *overlay*. Essa solução só foi elaborada considerando que a posição da câmera será estática ou com pouca variação ao longo de todo o processo de refinamento da massa. Caso essa hipótese se prove falsa, uma solução mais completa deverá ser implementada.

O funcionamento do script segue a lógica representada pelo seguinte algoritmo:

Algorithm 1 Aplicação de máscara

```
Carrega máscara como um vetor de valores RGB
Abre diretório de imagens input
for Cada imagem no diretório: do
    Carrega imagem como um vetor de valores RGB
    Soma os valores da máscara aos valores da imagem pixel a pixel
    Salva imagem processada
end for
```

Ou seja, o script inicialmente realiza a leitura da máscara e a armazena como um vetor de 3 dimensões (RGB) e de tamanho N, onde N é a quantidade de pixels na imagem (para uma imagen full HD, N = 2073600). Na sequência, é realizada dentro de um loop a leitura das imagens presentes no diretório de input uma a uma, carregando cada uma delas como uma matriz 3 x N. Para cada uma dessas imagens, são somados pixel a pixel os valores R, G e B da máscara, fazendo com que as regiões da máscara pintadas de branca sejam transpostas para a imagem final e as em preto não causem nenhuma alteração. Por fim, essa matriz é convertida para imagem e salva no formato .png.

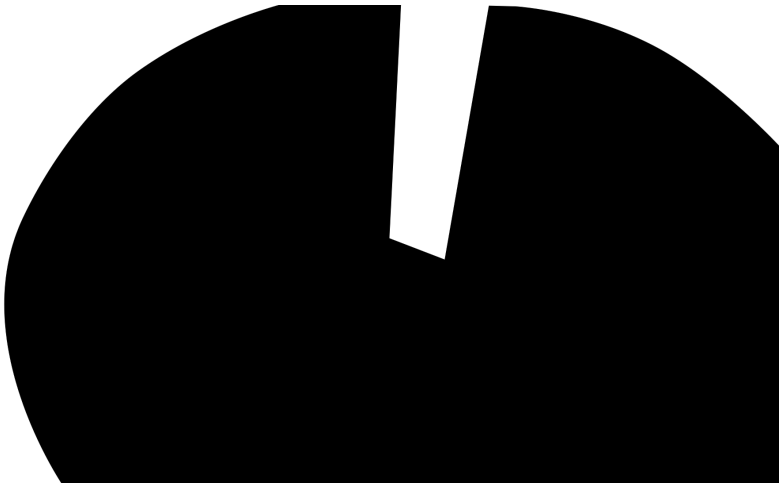

Figura 18: Máscara utilizada no pré-processamento das imagens

Antes de alcançar esta solução, o grupo tentou realizar a aplicação da máscara lendo a imagem da máscara como um vetor de luminosidade. Desta forma, os pixeis pretos da máscara indicariam que o valor do alpha deste mesmo pixel na imagem final seria 0 (transparente), enquanto os pixeis brancos indicariam um valor de alpha igual a 1, deixando a imagem inalterada. Ao concatenar os valores de luminosidade da máscara aos valores RGB da imagem, seria gerada uma imagem RGBA com a região de interessa da

imagem isolada. Porém, mesmo que esta versão da solução tenha gerado alterações visuais nas imagens, ela não conseguiu cumprir sua função dentro do treinamento do modelo por deixar os valores RGB da imagem inalterados. A biblioteca utilizada para a criação do modelo não interpreta valores de alpha, fazendo com que a máscara aplicada não fosse reconhecida durante o treinamento do modelo.

A lógica utilizada para criar a versão final deste script possui muito espaço para melhorias. Porém, dado o estado atual do projeto, acreditamos que a solução seja suficiente por hora. Uma possível melhoria, por exemplo, seria substituir a utilização de uma máscara criada manualmente por uma máscara automática que remova todos os pixels da imagem que não apresentem valores RGB dentro do espectro do marrom.

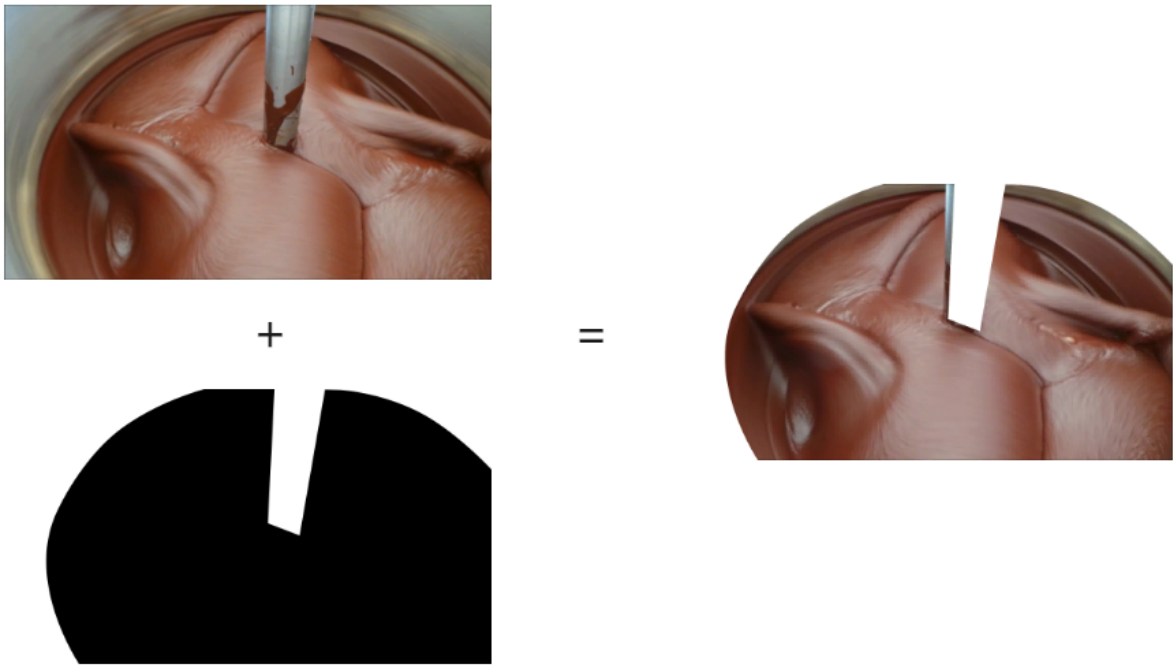

Figura 19: Frame antes e depois do pré-processamento

Esta etapa do desenvolvimento complementou o diagrama da solução, adicionando a ele mais uma etapa.

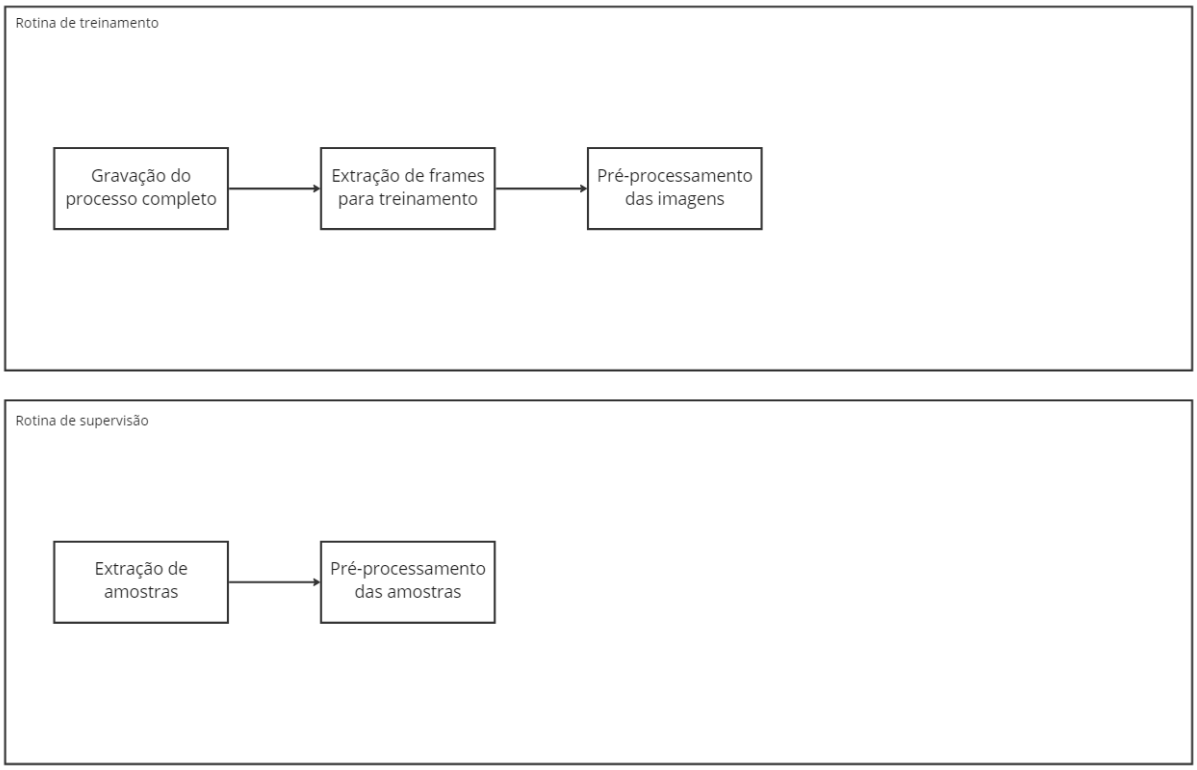

Figura 20: Estrutura da solução após desenvolvimento de script de pré-processamento de imagens

5.6 Classificação das imagens

Essa etapa do desenvolvimento teve por objetivo identificar o *frame* exato onde o processo de refino da massa se encerrou. A identificação do *frame* é crucial para o projeto pois é este que indica quais imagens representam uma massa ainda em refino (*frames* anteriores a ele) e quais indicam uma massa pronta (*frames* posteriores a ele), realizando assim a classificação necessária de todas as imagens para que possam ser utilizadas no treinamento da rede neural.

O grupo tentou combinar a identificação desse ponto através da coleta e análise de amostrar da massa em diversas marcações de tempo ao longo do processo de produção da leva teste de chocolate. Porém, como o projeto não conseguiu organizar a confecção da leva teste de chocolate, o grupo optou por realizar esta divisão de forma arbitrária num primeiro momento.

É fato que essa abordagem não reflete uma situação real, mas julgamos esta opção como a mais viável no momento, já que nos permitiu iniciar o desenvolvimento da rede neural sem depender de terceiros. Além disso, um retreinamento do modelo pode ser

facilmente realizado após a aquisição do banco de imagens definitivo.

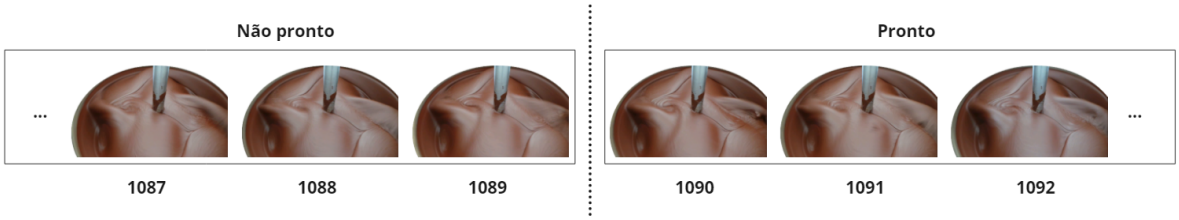

Figura 21: Ilustração da classificação dos frames

5.7 Criação da rede neural

Tendo todos os elementos necessários, o grupo iniciou o desenvolvimento da rede neural baseando-se no modelo Xception. Por se tratar de uma tarefa mais complexa que as anteriores, esta foi dividida em subetapas. Discutiremos as subetapas a seguir.

5.7.1 Geração dos *datasets*

Os *datasets* foram criados a partir das imagens pré-processadas da massa de cacau através de uma função da biblioteca Keras que se baseia na estrutura de diretórios em uma pasta para criar as classes e rotular as imagens.

Inicialmente o grupo tinha a intenção de desenvolver um modelo capaz de inferir o quão pronta está a massa, indicando essa métrica através de uma porcentagem. Porém, não foi encontrada uma abordagem simples para um problema dessa natureza. Por esse motivo, o propósito do modelo foi alterado para inferir se a massa está pronta ou não, caracterizando assim um problema de decisão binária.

Analisando o problema, pode-se interpretar as duas classes necessárias como as classes das massas "prontas"e a das massas "não prontas". Sendo assim, foi criado um diretório denominado "Massas"que contém as pastas com imagens de massas prontas (denominada "prontas") e não prontas (denominada "nao-prontas"). Assim como discutido anteriormente, essa divisão foi feita arbitrariamente e de forma manual, movendo as imagens pré-processadas da pasta de outuput do script de pré-processamento (denominada "raw") para as pastas que representem seus respectivos status.

Ao apontar a pasta "Massas"à função de criação de *datasets*, são criados automaticamente os *dataset* de treinamento e de validação com um *validation split* de 30% definido

pelo grupo, ou seja, 1188 imagens são separadas para o *dataset* de treinamento e 508 imagens são separadas para o de validação.

Figura 22: Plot das imagens que compõe o *dataset* de treinamento acompanhadas de suas respectivas classes. 0 Representa as massas que devem ser inferidas como não prontas e 1 as massas que devem ser inferidas como prontas.

5.7.2 Treinamento do modelo

Baseando-se no modelo Xception [4], o grupo realizou sua primeira tentativa de implementação da rede neural. O grupo optou por esta arquitetura pois, assim como é discutido em seu *paper*, apresentar algumas vantagens sobre o Inception V3 [5], modelo

amplamente utilizado para aplicações de reconhecimento de imagens onde é necessária uma rede neural convolucional.

Por meio da função disponibilizada pela própria biblioteca Keras [9], foi realizado um treinamento teste com 20 épocas utilizando-se o modelo Xception completo e, ao final do treinamento, o grupo encontrou alguns problemas na utilização do modelo.

O primeiro problema encontrado foi o tempo necessário para realizar o treinamento. Com apenas 20 épocas e um banco com 1696 imagens, o treinamento durou mais de cinco horas. Isso é um problema pois é estimado que o banco de imagens que utilizado em produção possuirá aproximadamente 20.000 imagens, estendendo ainda mais a duração desse treinamento.

O segundo problema é a quantidade de épocas necessárias para que o modelo convirja. O modelo não alcançou nenhum resultado significativo ao longo das 20 épocas, apresentando uma precisão de menos de 10% avaliando o *dataset* de treinamento e de 62% avaliando o *dataset* de validação.

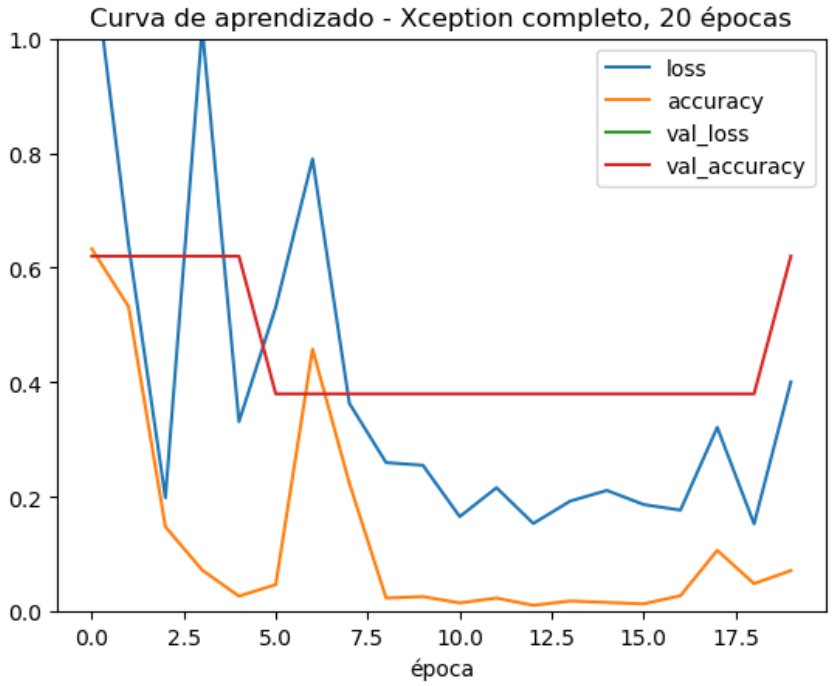

Figura 23: Curva de aprendizado do modelo Xception completo.

Além disso, por não possuir informações sobre a disponibilidade de computadores e GPUs para realizar o treinamento no ambiente de produção, o grupo optou por adotar uma versão simplificada do modelo Xception [10] onde é utilizada apenas uma versão adaptada do fluxo de entrada do modelo Xception, uma camada totalmente conectada

na saída de dimensão 1 com ativação *sigmoid* e um tamanho reduzido de imagem (180 x 180).

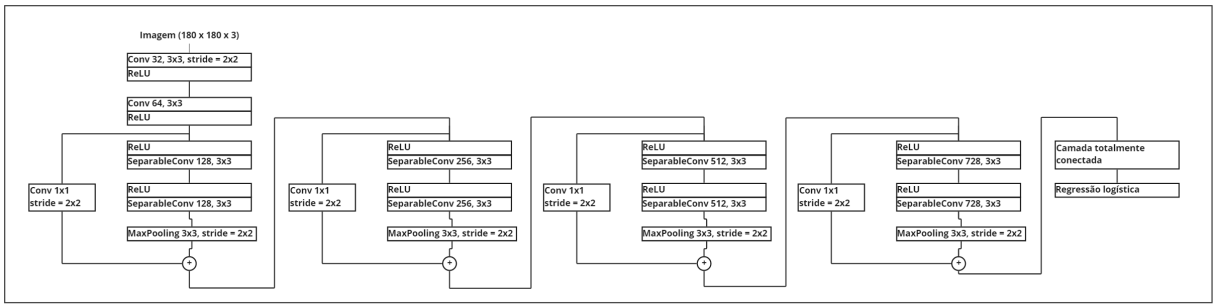

Figura 24: Arquitetura do modelo Xception simplificado

Optamos por remover o fluxo do meio da arquitetura Xception por conta da quantidade de camadas convolucionais nele criada, reduzindo assim a quantidade de parâmetros treináveis da rede em 86.67%, passando de 20.811.050 para 2.773.913. Além disso, consideramos esta uma boa abordagem ao problema por se ater aos princípios do modelo Xception, desacoplando o mapeamento das correlações entre canais das correlações espaciais.

Criado o código, foi feito mais um treinamento teste com o mesmo *dataset* de treinamento e 20 épocas, gerando os seguintes resultados.

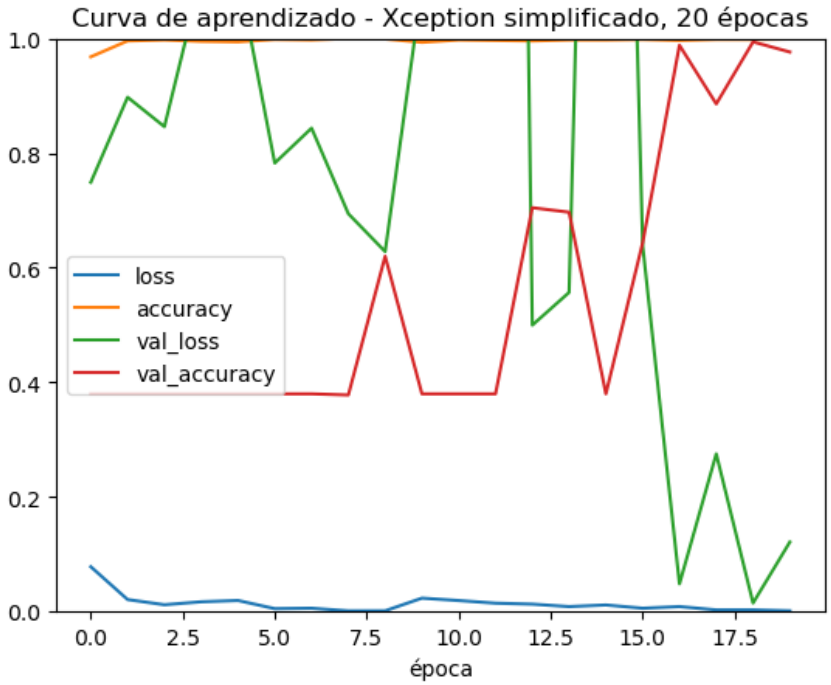

Figura 25: Curva de aprendizado do modelo Xception simplificado com 20 épocas

Pode-se observar uma convergência mais rápida do que no modelo completo, alcançando uma precisão de 97.24% ao avaliar o *dataset* de validação (adotando um limiar de decisão de 50%) quando submetido ao mesmo treinamento. A duração do treinamento também chamou a atenção do grupo, durando menos de uma hora.

Visto que o modelo simplificado foi capaz de adquirir uma boa precisão com um treinamento de 20 épocas, o grupo tentou aumentar a quantidade de épocas utilizadas, refazendo o treinamento do modelo com 60 épocas e obtendo os seguintes resultados.

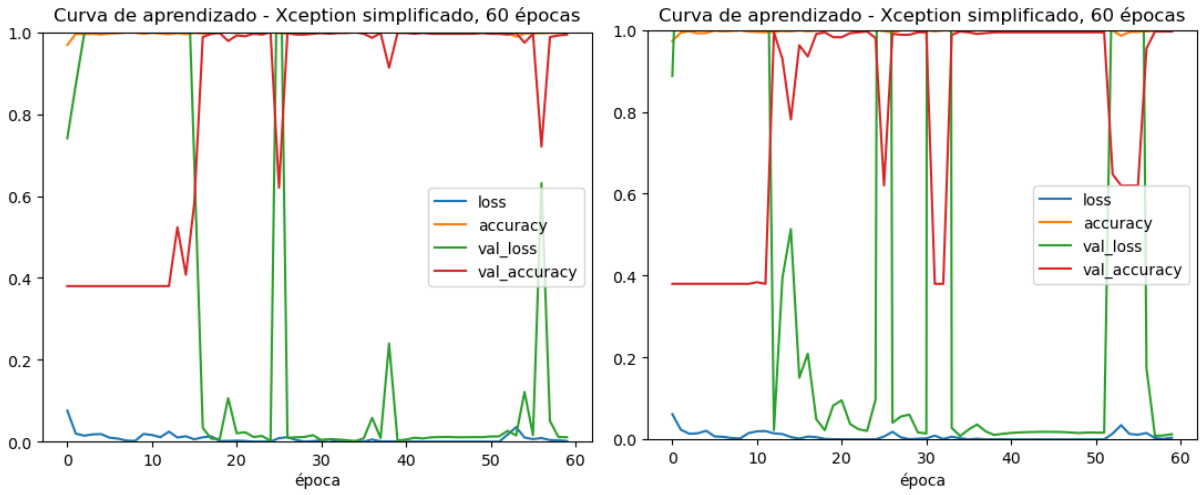

Figura 26: Curvas de aprendizado do modelo Xception simplificado com 60 épocas.
Para analisar a ocorrência dos vales de precisão, o treinamento foi feito duas vezes.

N = 508	**Inferiu pronto**	**Inferiu não pronto**
Pronto	**313**	**2**
Não pronto	**0**	**193**

Tabela 1: Matriz de confusão do primeiro treinamento

N = 508	**Inferiu pronto**	**Inferiu não pronto**
Pronto	**312**	**3**
Não pronto	**1**	**192**

Tabela 2: Matriz de confusão do segundo treinamento

É possível observar a presença de vales periódico na precisão acompanhados de picos de erro no *dataset* de validação. O grupo inicialmente levantou a hipótese de que os picos representam um crescente *overfitting* sendo desenvolvido pela rede, mas a hipótese foi abandonada pois a precisão volta a crescer após algumas épocas.

O grupo concluiu que a ocorrência dos picos de erro e dos vales de precisão está atrelada à função de custo utilizada e à taxa de aprendizado do modelo, definidos como Adam e 0.001 respectivamente. Por esse motivo, foi feito um terceiro treinamento utilizando-se uma taxa de aprendizado de 0.000001. Além disso, observa-se um período de estabilidade comum aos dois gráficos entre as épocas 40 e 50. Assim, o terceiro treinamento também alterou a quantidade de épocas utilizadas para 45.

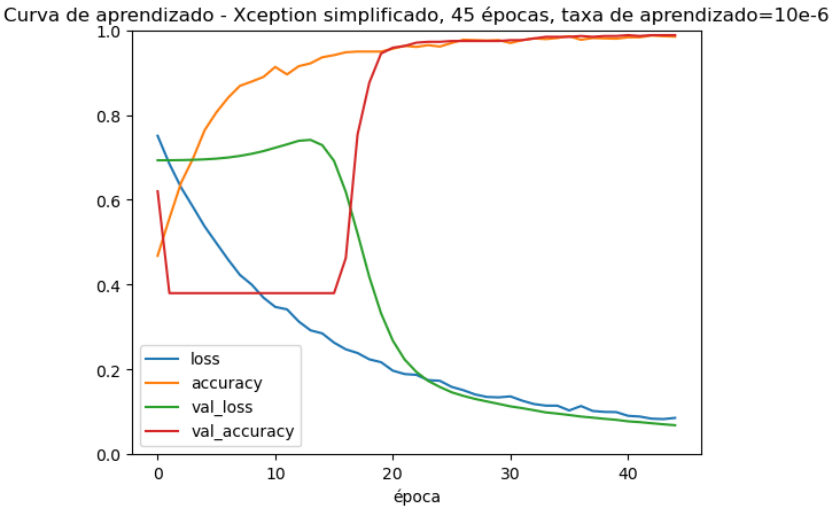

Figura 27: Curvas de aprendizado do terceiro treinamento.
A precisão máxima alcançada foi 98.82% a partir da 41ª época.

N = 508	**Inferiu pronto**	**Inferiu não pronto**
Pronto	**309**	**6**
Não pronto	**0**	**193**

Tabela 3: Matriz de confusão do terceiro treinamento

Apesar de apresentar uma curva de aprendizado muito mais regular, o terceiro treinamento não gerou uma melhor precisão, estabilizando-se com uma precisão de 98.82%. Por esse motivo, o grupo seguiu a análise sobre o primeiro e segundo treinamentos.

Analisando as matrizes de confusão, foras extraídas as seguintes métricas.

	Primeiro treinamento	**Segundo treinamento**
Acurácia	**99.61%**	**99.21%**
Precisão	**100%**	**99.68%**
Revocação	**99.36%**	**99.05%**
F1	**99.68%**	**99.36%**

Tabela 4: Métricas extraídas das matrizes de confusão

Através das métricas, é possível concluir que o modelo apresentou um ótimo desempenho ao ser treinado por 60 épocas e com taxa de aprendizado de 0.001. A alta acurácia indica que a performance geral do modelo é boa e os demais parâmetros indicam que o modelo não tende a cometer erros do tipo I e II. Porém, a precisão chamou a atenção do grupo.

Os falsos positivos representam um problema crítico no projeto pois sua ocorrência faria com que o processo de refino da massa de cacau se encerrasse com a massa ainda não pronta, estragando a leva de chocolate. Por conta do falso positivo observado nas matrizes de confusão, o grupo optou por adicionar medidas paliativas que evitem a ocorrência do problema.

A medida paliativa consiste em alterar a forma como a supervisão da massa acontece. Originalmente, o grupo planejou realizar a supervisão da massa capturando uma foto da massa a cada 10 minutos e alimentando esta foto ao modelo, que realizaria a inferência e ditaria se o processo deve continuar ou não. Para evitar os falsos positivos, passou-se a capturar múltiplas fotos ao longo do intervalo de 10 minutos, assim como mencionado na seção 6.2.

Para definir a quantidade de fotos necessárias para evitar os falsos positivos, foi feita uma análise estatística. Podemos modelar a predição do modelo como um evento binário de probabilidade de acerto média (levando em consideração os dois treinamentos realizados) de 99.41%. Com essas informações, foram traçados os gráficos da função distribuição de probabilidades de alguns cenários de acordo com a equação da distribuição binomial ilustrada a seguir.

$$p_X(x) = P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$$

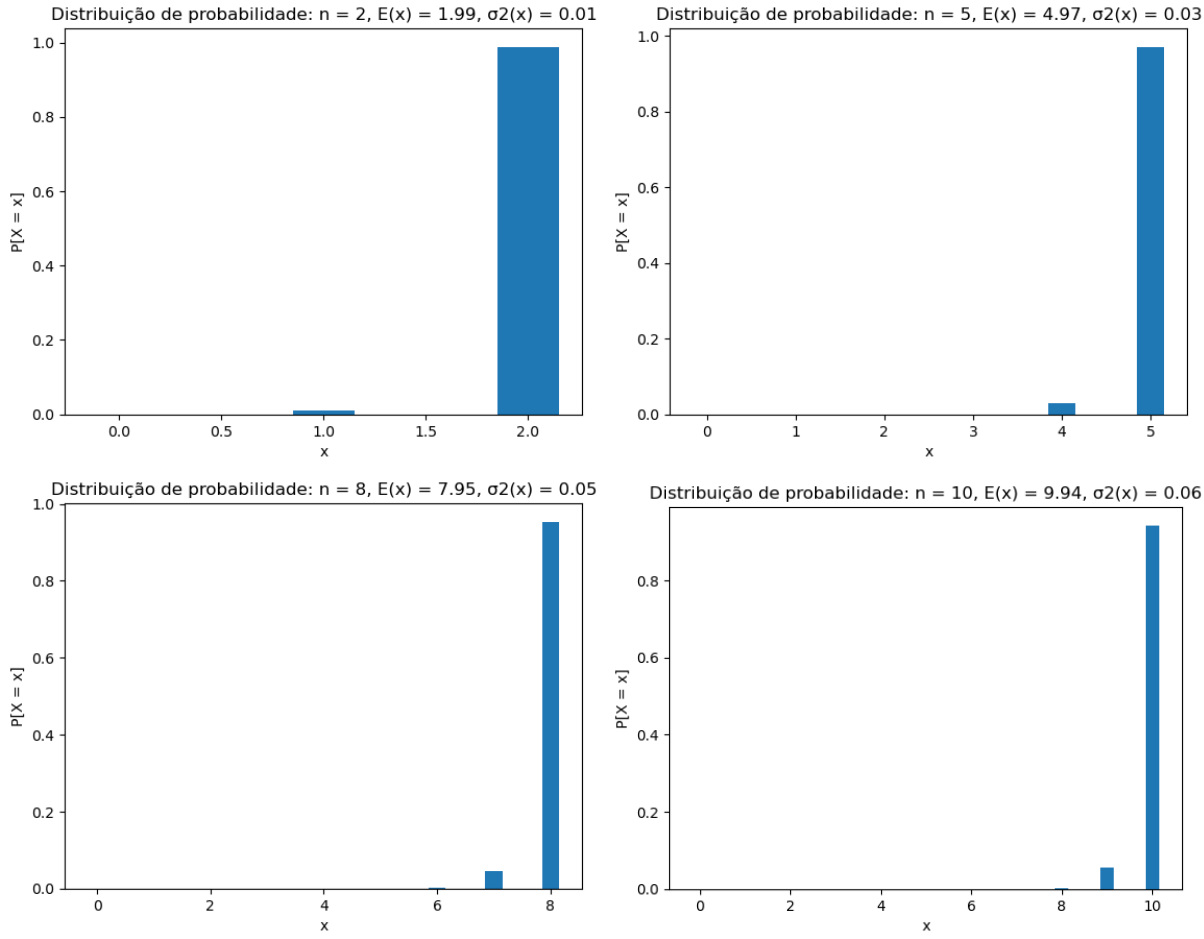

Figura 28: Gráficos da distribuição de probabilidade do modelo considerando 2, 5, 8 e 10 tentativas. O eixo Y indica a probabilidade do modelo acertar exatamente x vezes.

Analisando os gráficos, percebeu-se que o modelo tem alta probabilidade de acertar a maioria das inferências para quantidades de tentativa maiores do que 2 por conta da alta acurácia. Sendo assim, o grupo optou por capturar 10 fotos ao longo dos 10 minutos de intervalo. Esse parâmetro pode facilmente ser alterado no futuro e pode ser usado pra contornar eventuais problemas de desempenho do modelo no ambiente de produção.

Por fim, definido a arquitetura da rede, o grupo criou um script responsável por executar as rotinas de treinamento e supervisão, finalizando assim o software. O script simplesmente implementam uma interface no terminal que permite que o usuário escolha a rotina a ser executada.

Ao final desta etapa do projeto, o diagrama da solução do projeto foi completo.

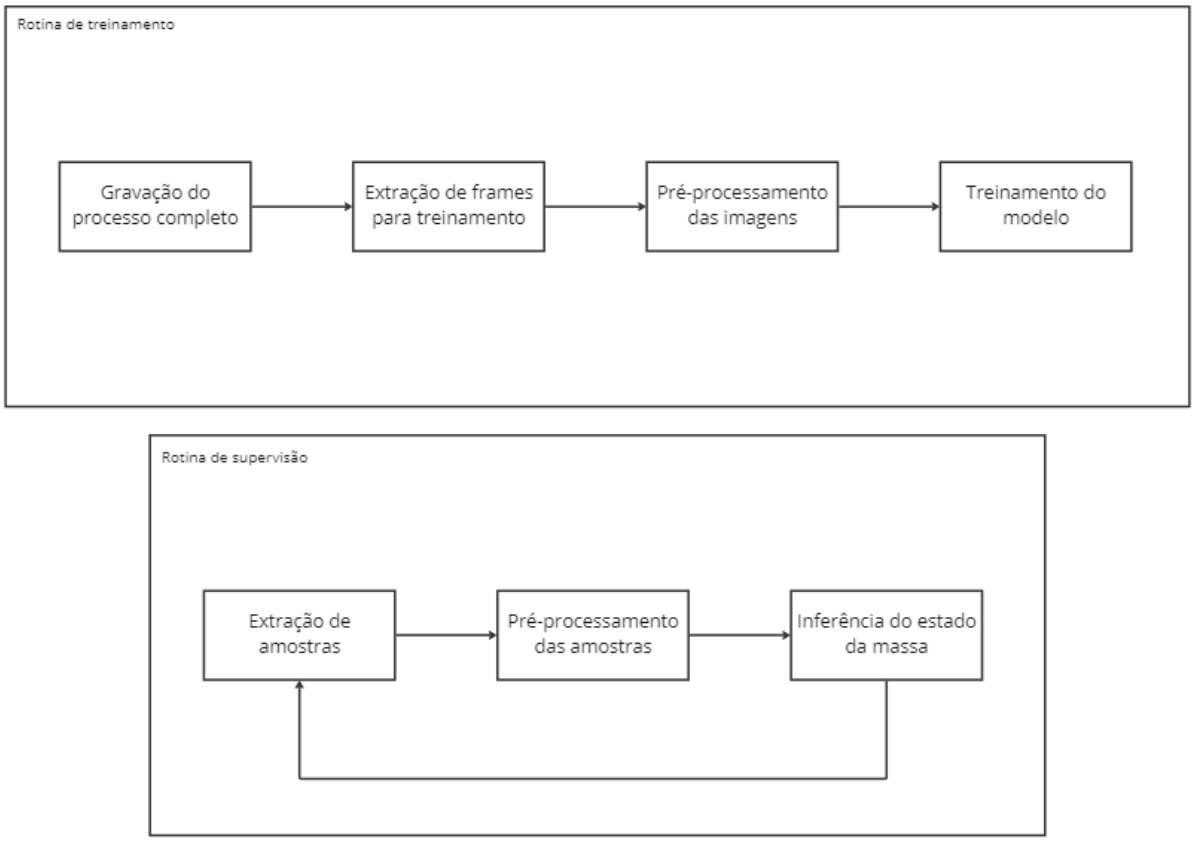

Figura 29: Estrutura da solução após desenvolvimento do modelo

5.7.3 *Data augmentation*

O grupo foi capaz de gerar um banco com uma quantidade suficiente de imagens para realizar o treinamento do modelo através da extração de *frames* de vídeos da internet, mas mesmo sendo suficiente, a quantidade não foi ideal. Por conta disso, foi considerada a opção de introduzir no programa uma etapa de *data augmentation* responsável por adicionar artificialmente ligeiras transformações nas imagens com o objetivo de diversificar as amostras. Dentre essas transformações, foram cogitados espelhamentos verticais, espelhamentos horizontais e ligeiras rotações aleatórias. Vale ressaltar que esta etapa não introduz novas amostras à base.

Porém, em conversas posteriores, o grupo questionou a eficácia de adicionar esta diversidade artificial à amostra. Em sua versão final, o sistema deve possuir uma câmera estática dentro da *melanger* que sempre capturará imagens do mesmo ângulo e orientação. Assim, ao adicionar imagens artificiais, estaríamos fornecendo ao modelo imagens que ele não terá acesso no futuro, prejudicando assim a avaliação e possivelmente o desempenho do modelo.

Pensando nisso, o grupo realizou testes treinando o modelo por 18 épocas utilizando um *dataset* com transformações artificiais e outro sem. Seguem os resultados.

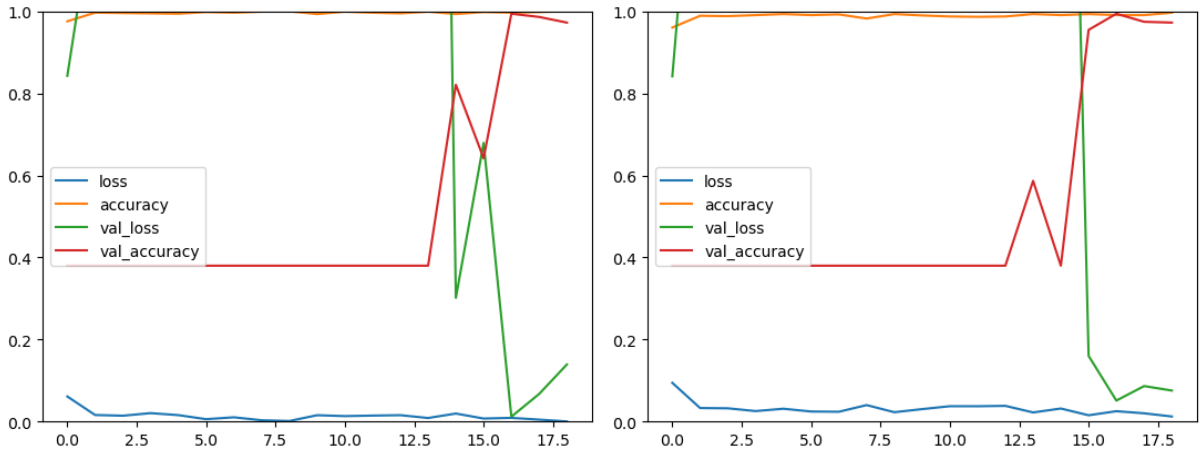

Figura 30: Curvas de aprendizado do modelo **sem** *data augmentation* (esquerda) e **com** *data augmentation* (direita)

N = 508	**Inferiu pronto**	**Inferiu não pronto**
Pronto	**301**	**14**
Não pronto	**0**	**193**

Tabela 5: Matriz de confusão do modelo **sem** *data augmentation*

N = 508	**Inferiu pronto**	**Inferiu não pronto**
Pronto	**302**	**13**
Não pronto	**1**	**192**

Tabela 6: Matriz de confusão do modelo **com** *data augmentation*

Pode-se observar que a introdução de transformações artificiais aumenta ligeiramente o erro e diminui ligeiramente a acurácia do treinamento. Além disso, utilizando-se o modelo treinado com *data augmentation*, o modelo cometeu um erro tipo falso positivo a mais do que o modelo treinado sem *data augmentation*. Porém, não se soube concluir se essa diferença está atrelada aos parâmetros controlados no teste.

Através dos resultados, pode-se concluir que a adição de transformações artificiais não impacta significantemente a precisão do modelo, mostrando-se assim uma adição desnecessária ao projeto.

6 CONSIDERAÇÕES FINAIS

6.1 Conclusão

Levando em consideração todas as métricas analisadas observadas no tópico anterior, o grupo concluiu que o modelo que deve ser aplicado em produção será treinado por 60 épocas utilizando-se uma taxa de aprendizado de 0.001, e assim o projeto conseguiu cumprir seus objetivos. A metodologia abordada, mesmo com algumas alterações ao longo do desenvolvimento da solução, foi capaz de gerar um software funcional que fornece uma maneira automatizada de realizar a supervisão da massa de cacau em produção.

O modelo da rede neural foi capaz de cumprir sua função com alta acurácia, fazendo com que o software cumpra seus requisitos funcionais. As adaptações realizadas ao modelo Xception foram capazes de diminuir a quantidade de parâmetros treináveis da rede, diminuindo o tempo necessário para seu treinamento, e os scripts de aquisição de imagens foram capazes de gerar bancos de imagens automaticamente.

Uma característica interessante do projeto é a sua flexibilidade. Por ser capaz de criar bancos de imagens e treinar uma rede neural automaticamente, a aplicação do software não se limita ao escopo para o qual foi originalmente planejado, sendo uma opção para qualquer problema que exija uma supervisão contínua de um elemento. Dentro do contexto de indústria 4.0, essa se mostra uma característica extremamente vantajosa.

Com as mudanças de escopo que ocorreram ao londo do projeto, os requisitos de confiabilidade e manutenibilidade definidos na seção 4 não puderam ser implementados em sua totalidade até o presente momento. Como o grupo não conseguiu realizar testes no equipamento, não foi possível estudar a frequência de falhas dos equipamentos, se tornando um ponto a ser estudado ao longo dos próximos passos do projeto. Além disso, por não saber em que equipamento o software será hospedado, a manutenibilidade também não pode ser verificada.

6.2 Contribuições

O grupo recebeu algumas contribuições ao longo do projeto, sendo grande parte delas intermediada pela orientadora. A reunião com o integrante da empresa NEC foi extremamente importante por ajudar o grupo a comparar a solução levantada com outras soluções existentes no mercado. A reunião com o chocolateiro do projeto Amazônia 4.0 também foi necessária para que o grupo começasse a entender o processo produtivo do chocolate. Os códigos e scripts desenvolvidos ao longo do projeto foram criados em sua totalidade pelos integrantes do grupo, sem contribuições externas.

6.3 Perspectivas de continuidade

Sendo uma solução projetada para o Amazônia 4.0, existe a perspectiva já confirmada com um dos integrantes do projeto de que a solução seja utilizada nas linhas de produção de chocolate do projeto. Por esse motivo, o grupo registrou todos os pontos importantes que devem ser atacados futuramente.

A primeira e principal questão que deve ser abordada é justamente a aplicação da solução no ambiente de produção. Por questões de agenda já discutidas anteriormente, o grupo não conseguiu agendar testes presenciais com a empresa Omega7, o que fez com que especificidades do ambiente de produção como iluminação, umidade, temperatura, capacidade de processamento de máquinas, disponibilidade de microcontroladores e GPUs não fossem levados em consideração ao longo do desenvolvimento da solução. Por um lado, essa incerteza levou o grupo a desenvolver uma solução sucinta e facilmente adaptável, o que é uma vantagem. Porém, isso também privou o grupo de muitos detalhes potencialmente importantes para o funcionamento da solução. É seguro dizer que alguns problemas surgirão ao aplicara a solução em produção e que o projeto está preparado para sofrer qualquer alteração necessária.

Um ponto que preocupou o grupo ao longo do desenvolvimento do projeto foi o desempenho do modelo quando aplicado em produção. Mesmo que este tenha empenhado extremamente bem ao longo dos testes, não se pode afirmar que seu desempenho será igual quando em produção. Porém, como o modelo foi capaz de distinguir as massas de cacau prontas e não prontas ao longo dos testes, o grupo está seguro de que ele também será capaz de empenhar esse papel em produção. Mas caso sua acurácia acabe não sendo alta, o script de inferência do estado da massa pode ser alterado para aumentar a quantidade de amostras averiguadas de uma só vez (atualmente definida como 10 pelo grupo).

O último ponto que foi removido do escopo deste trabalho e que deverá ser discutido junto dos integrantes da mepresa Omega7 é a forma como a solução será aplicada em produção. Nos planejamentos iniciais do grupo, foi planejado utilizar um microcontrolador Raspberry para rodar a rotina de supervisão do software e uma máquina com maior capacidade de processamento para realizar a rotina de treinamento.

REFERÊNCIAS

1 MELLO, A. F. de. *Dilemas e desafios do desenvolvimento sustentável da Amazônia: O caso brasileiro*. Disponível em: ⟨https://journals.openedition.org/rccs/6025⟩. Acesso em: 03 Jul. 2022.

2 UOL. *Ranking Universitário*. Disponível em: ⟨https://ruf.folha.uol.com.br/2019/ranking-de-universidades/principal/⟩. Acesso em: 03 Jul. 2022.

3 BOND, L. *Indústria de alimentos e bebidas faturou R$ 699,9 bi em 2019*. Disponível em: ⟨https://agenciabrasil.ebc.com.br/economia/noticia/2020-02/industria-de-alimentos-e-bebidas-faturaram-r-6999-bi-em-2019⟩. Acesso em: 03 Jul. 2022.

4 CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. Disponível em: ⟨http://arxiv.org/abs/1610.02357⟩.

5 SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. Disponível em: ⟨http://arxiv.org/abs/1512.00567⟩.

6 BELLARD, G. *FFmpeg*. Disponível em: ⟨https://ffmpeg.org/about.html⟩. Acesso em: 18 Out. 2022.

7 NEC. Disponível em: ⟨https://www.nec.com⟩. Acesso em: 10 Jul. 2022.

8 MACHINERY, A. *Carregando a batedeira para 50 kg de chocolate*. Disponível em: ⟨https://www.youtube.com/watch?v=dUM0qDohHnI&ab_channel=AllureMachinery⟩. Acesso em: 09 Out. 2022.

9 CHOLLET, F. *Xception*. Disponível em: ⟨https://keras.io/api/applications/xception/⟩. Acesso em: 09 Nov. 2022.

10 CHOLLET, F. *Image classification from scratch*. 2020. Disponível em: ⟨https://keras.io/examples/vision/image_classification_from_scratch/⟩. Acesso em: 21 Jul. 2022.

11 ACADEMY, D. S. *Capítulo 44 – Reconhecimento de Imagens com Redes Neurais Convolucionais em Python – Parte 1*. Disponível em: ⟨https://www.deeplearningbook.com.br/reconhecimento-de-imagens-com-redes-neurais-convolucionais-em-python-parte-1/⟩. Acesso em: 10 Jul. 2022.

12 LECUN yann. *Hierarchical Models Of Perception and Reasoning*. Disponível em: ⟨http://matt.colorado.edu/compcogworkshop/talks/lecun.pdf.⟩ Acesso em: 10 Jul. 2022.

13 YAN, Q. et al. Deep hdr imaging via a non-local network. *IEEE Transactions on Image Processing*, v. 29, p. 4308–4322, 2020.

14 RUSSAKOVSKY JIA DENG, H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. O.; FEI-FEI, L. Deep hdr imaging via a non-local network. *International Journal of Computer Vision*, 2015.

15 SEACHAOS. *Get Heatmap from CNN (Convolution Neural Network), AKA CAM*. Disponível em: ⟨https://tree.rocks/get-heatmap-from-cnn-convolution-neural-network-aka-grad-cam-222e08f57a34⟩. Acesso em: 27 Nov. 2022.

16 PANDEY, A. *Depth-wise Convolution and Depth-wise Separable Convolution*. Disponível em: ⟨https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec⟩. Acesso em: 23 Nov. 2022.