UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

TIAGO MESQUITA DE OLIVEIRA

INFLUÊNCIA DA QUANTIZAÇÃO DE REDES NEURAIS CONVOLUCIONAIS NA CLASSIFICAÇÃO DE IMAGENS DE ABELHAS PORTADORAS DE PÓLEN

FORTALEZA

2025

TIAGO MESQUITA DE OLIVEIRA

INFLUÊNCIA DA QUANTIZAÇÃO DE REDES NEURAIS CONVOLUCIONAIS NA CLASSIFICAÇÃO DE IMAGENS DE ABELHAS PORTADORAS DE PÓLEN

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Sistemas de Informação

Orientador: Prof. Dr. José Maria da Silva Monteiro Filho

Coorientador: Prof. Dr. José Wellington Franco da Silva

FORTALEZA

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

O52i Oliveira, Tiago Mesquita de.
Influência da quantização de redes neurais convolucionais na classificação de imagens de abelhas portadoras de pólen / Tiago Mesquita de Oliveira. – 2025.
60 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2025.
Orientação: Prof. Dr. José Maria da Silva Monteiro Filho.
Coorientação: Prof. Dr. José Wellington Franco da Silva.

1. Abelhas portadoras de pólen. 2. Aprendizado profundo. 3. Quantização. I. Título.
CDD 005

TIAGO MESQUITA DE OLIVEIRA

INFLUÊNCIA DA QUANTIZAÇÃO DE REDES NEURAIS CONVOLUCIONAIS NA CLASSIFICAÇÃO DE IMAGENS DE ABELHAS PORTADORAS DE PÓLEN

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Sistemas de Informação

Aprovada em: 28/11/2025

BANCA EXAMINADORA

Prof. Dr. José Maria da Silva Monteiro Filho (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Wellington Franco da Silva (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. João Paulo do Vale Madeiro
Universidade Federal do Ceará (UFC)

Prof. Dr. Renato William Rodrigues de Souza
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

À minha mãe, minha família e todos que contribuíram na realização deste trabalho.

AGRADECIMENTOS

Agradeço à minha família por me dar o suporte e incentivos necessários aos meus estudos.

Ao Prof. Dr. José Maria da Silva Monteiro Filho pela confiança, orientação, apoio, experiência e conhecimentos compartilhados comigo durante o mestrado. Além disso, agradeço sua ajuda na escrita dessa dissertação, na análise, nos testes e na validação dos códigos e resultados deste trabalho.

Ao Prof. Dr. José Wellington Franco da Silva pela orientação e ajuda com o assuntos de visão computacional, modelos de classificação, na revisão dessa dissertação e na análise, testes e validação dos códigos e resultados.

A colega Letícia Torres pela ajuda na análise, testes e validação dos códigos e resultados e na condução dos testes contidos na seção 6.2

Aos professores do programa de mestrado pela dedicação e zelo no repasse de conhecimentos. Aos professores participantes da banca examinadora pela disponibilidade e sugestões.

Aos amigos do programa de mestrado pelo apoio, amizade e ajuda durante as disciplinas.

“O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.”

(José de Alencar)

RESUMO

O reconhecimento de abelhas portadoras de pólen pode fornecer informações importantes sobre as condições de funcionamento das colmeias. Pode indicar também a intensidade da atividade de polinização, a qual é um aspecto fundamental para muitas espécies de plantas e de grande interesse comercial para a agricultura. Neste contexto, sistemas embarcados são comumente utilizados no processo de aquisição e processamento de imagens de abelhas nas entradas das colmeias. Todavia, esses sistemas, em geral, possuem restrições quanto à capacidade computacional e ao espaço de armazenamento. Neste trabalho, foi analisado o desempenho de quatorze modelos de redes neurais convolucionais profundas na solução do problema de classificação de abelhas portadoras e não portadoras de pólen a partir de um conjunto de imagens. Também foi analisado como o processo de quantização influi nesses resultados. A quantização permite diminuir o tempo de inferência e o tamanho dos modelos devido ao fato de realizar cálculos e armazenar os números em uma estrutura de menor precisão, o que pode ser vantajoso em contextos baseados na utilização de sistemas embarcados. Os resultados experimentais apontam que é possível reduzir o tempo de inferência e/ou o tamanho do modelo sem diminuir o desempenho das medidas de acurácia, precisão, revocação, *F1-score* e taxa de falso positivo, facilitando a implantação em um sistema embarcado. O melhor resultado foi obtido pela rede AlexNet quantizada para *int8*, reduzindo o tempo de inferência e o tamanho do modelo em, respectivamente, 45.63% e 71.73%.

Keywords: abelhas portadoras de pólen; aprendizado profundo; quantização.

ABSTRACT

The recognition of pollen-carrying bees can provide important information about the operational conditions of beehives. It can also indicate the intensity of pollination activity, which is a fundamental aspect for many plant species and of great commercial interest to agriculture. In this context, embedded systems are commonly used in the process of acquiring and processing images of bees at the entrances of beehives. However, these systems generally have constraints regarding computational capacity and storage space. In this work, the performance of fourteen deep convolutional neural network models in solving the problem of classifying pollen-bearing and non-pollen-bearing bees from a set of images was analyzed. The study also analyzed how the quantization process influences these results. Quantization allows for a reduction in inference time and model size by performing calculations and storing numbers in a lower-precision format, which can be advantageous in contexts based on the use of embedded systems. The experimental results indicate that it is possible to reduce the inference time and/or the model size without diminishing the performance metrics of accuracy, precision, recall, F1-score, and false positive rate, facilitating deployment in an embedded system. The best result was achieved by the AlexNet network quantized to int8, reducing the inference time and model size by 45.63% and 71.73%, respectively.

Keywords: pollen bearing bees; deep learning; quantization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Network* / Rede Neural Convolucional
CPU	*central processing unit* / unidade central de processamento
DNN	*Deep Convolutional Neural Networks* / Redes Neurais Convolucionais Profundas
FAIR	*Facebook Artificial Intelligence Research*
GB	*gigabyte*
GPU	*graphics processing unit* / unidade de processamento gráfico
IoT	*Internet of Things* / Internet das Coisas
KB	*kilobyte*
KNN	*K-Nearest Neighbors* / K-vizinhos mais próximos
NAS	*neural architecture search* / pesquisa de arquiteturas de rede
PCA	*Principal Component Analysis*
RAM	*random access memory*
RGB	*red, green, blue*
ROC	*Receiver Operating Characteristic*
ReLU	*rectified linear unit* / unidade linear retificada
SVM	*Support Vector Machine* / Máquina de Vetores de Suporte
VGG	*Visual Geometry Group*
VRAM	*video random access memory*
faster RCNN	*Faster Region-based Convolutional Neural Network*
float16	*16-bit floating point number* / número de ponto flutuante de 16 *bits*
float32	*32-bit floating point number* / número de ponto flutuante de 32 *bits*
int8	*8-bit integer* / inteiro de 8 *bits*
IA	inteligência artificial
TFP	taxa de falso positivo

SUMÁRIO

1 INTRODUÇÃO

As abelhas constituem um dos principais vetores de polinização, pois transferem pólen entre flores, o que auxilia o processo de fertilização Bilik *et al.* (2024), Hadjur *et al.* (2022). A polinização desempenha um papel crucial na preservação do meio ambiente, pois contribui substancialmente para a reprodução de uma ampla variedade de plantas com flores, incluindo culturas agrícolas (MONTEIRO *et al.*, 2021). A produção agrícola que depende de animais polinizadores, como as abelhas, quadruplicou nas últimas cinco décadas (FOOD; NATIONS, 2016). Desta forma, pode-se afirmar que as abelhas são cruciais para a agricultura e, consequentemente, para a existência humana (RODRIGUEZ *et al.*, 2018). As abelhas também produzem mel, geleia real, própolis, pólen e cera, substâncias que oferecem diversos benefícios à saúde dos seres humanos (HADJUR *et al.*, 2022).

Por outro lado, as abelhas apresentam um comportamento social complexo, caracterizado por hierarquia, papéis bem definidos, horários de atuação e interações variadas. Para compreender esse comportamento, os cientistas realizam observações cuidadosas e registros precisos (RODRIGUEZ *et al.*, 2018). Inspecionar as atividades de abelhas produtoras de pólen pode ser interessante não só para apicultores, mas também para agricultores, ecologistas e biólogos (ZACEPINS *et al.*, 2015). Para os apicultores, os indicadores mais importantes da saúde de uma colmeia são o número de abelhas melíferas que saem e entram em uma colmeia, e o número de abelhas melíferas que transportam pólen (STOJNIĆ *et al.*, 2018). Em geral, o cálculo desses indicadores é realizado por especialistas de forma manual. Todavia, essa é uma tarefa demorada e dispendiosa, a qual requer longos períodos de observação e, por vezes, conhecimentos específicos (RODRIGUEZ *et al.*, 2018). Em (BERKAYA *et al.*, 2021), os autores relataram a extrema dificuldade de obter registros precisos por meio de contagens realizadas na entrada da colmeia. Uma forma diferente de inspeção consiste em abrir a colmeia com a finalidade de verificar o seu aspecto. Contudo, essa abordagem pode levar os insetos a situações de estresse, o que pode comprometer a saúde da colmeia (SLEDEVIČ, 2018). Por estas razões, torna-se necessário desenvolver estratégias de observação menos intrusivas e mais eficazes.

Nos últimos anos, com o desenvolvimento de sistemas embarcados, redes de sensores e internet das coisas, a utilização de recursos tecnológicos para a observação de colmeias tornou-se viável, facilitando a detecção e o registro do comportamento das abelhas (RODRIGUEZ *et al.*, 2018; YANG; COLLINS, 2019). A utilização de técnicas de visão computacional e inteligência artificial (IA) permite não só adquirir e processar imagens da entrada de uma determinada

colmeia, mas também reconhecer a presença de abelhas portadoras de pólen nas imagens capturadas e computar automaticamente indicadores acerca da saúde da colmeia (SLEDEVIČ, 2018; HADJUR *et al.*, 2022). Neste cenário, diversos equipamentos de observação e monitoramento de colmeias têm sido propostos. Entretanto, a grande maioria desses equipamentos possui reduzida capacidade de armazenamento e processamento, o que pode inviabilizar a utilização de modelos preditivos mais complexos, como as redes neurais profundas, por exemplo. Para enfrentar esse desafio, uma estratégia pode ser a utilização de modelos quantitativos. A quantização permite reduzir o tempo de inferência e o tamanho dos modelos ao realizar cálculos e armazenar os números em uma estrutura de menor precisão, o que pode ser vantajoso em contextos de uso de sistemas embarcados. Adicionalmente, essa abordagem possibilita a utilização de dispositivos mais modestos e baratos, além de reduzir o consumo de energia, o que aumenta a autonomia da bateria dos equipamentos utilizados.

1.1 Objetivos

Esta dissertação tem como objetivo principal avaliar o impacto da quantização em redes neurais convolucionais profundas, capazes de classificar abelhas portadoras e não portadoras de pólen a partir de imagens.

Entre os objetivos específicos estão:

- A análise e a comparação dos tempos de inferência e dos tamanhos dos modelos.
- A análise e a comparação dos tamanhos dos modelos.

1.2 Contribuições

Seguem as principais contribuições deste trabalho:

- Avaliação de quatorze modelos de redes neurais convolucionais profundas na classificação de abelhas portadoras de pólen versus aquelas sem pólen em imagens capturadas na entrada de uma determinada colmeia;
- Análise do impacto do processo de quantização no tempo de inferência desses modelos;
- Análise do impacto do processo de quantização no tamanho dos modelos preditivos, mais precisamente, das redes neurais convolucionais profundas.

1.3 Organização da dissertação

Esta dissertação está organizada em sete capítulos, descritos brevemente a seguir:

- **Capítulo 1. Introdução**
  No primeiro capítulo, apresenta-se uma visão geral do problema de classificação de abelhas portadoras de pólen versus aquelas sem pólen, com base em imagens capturadas na entrada de uma determinada colmeia.
- **Capítulo 2. Fundamentação Teórica**
  No segundo capítulo são apresentados os principais conceitos utilizados neste trabalho.
- **Capítulo 3. Redes Convolucionais Avaliadas**
  O terceiro capítulo aborda alguns conceitos relacionados aos modelos de classificação utilizados neste trabalho.
- **Capítulo 4. Trabalhos Relacionados**
  Este capítulo descreve e compara os principais trabalhos da literatura.
- **Capítulo 5. Metodologia Experimental**
  Neste capítulo, são descritos os experimentos realizados para avaliar o impacto da quantização no problema investigado.
- **Capítulo 6. Resultados e Discussão**
  Neste capítulo são apresentados e discutidos os resultados obtidos.
- **Capítulo 7. Conclusão e Trabalhos Futuros**
  Finalmente, neste capítulo, são destacadas as principais conclusões obtidas e apontadas as direções para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos fundamentais abordados neste trabalho. Inicialmente, será discutido o funcionamento das redes neurais convolucionais. Posteriormente, será apresentada a definição de transferência de aprendizado. Finalmente, o conceito de quantização de redes neurais convolucionais é apresentado e discutido.

2.1 Redes Neurais Convolucionais

De acordo com IBM (2017), o aprendizado de máquina (ou aprendizado automático, ou ainda *machine learning*, em inglês) pode ser definido como a capacidade de uma máquina de imitar comportamento inteligente por meio de algoritmos que podem "aprender" os padrões presentes em dados de treinamento sem serem programados explicitamente. Arthur Samuel, em 1959, definiu o aprendizado de máquina como o campo de estudo cuja finalidade é fornecer aos computadores a habilidade de aprender sem serem programados explicitamente (SAMUEL, 1959). Mais formalmente, Tom Mitchell, em 1997, conceituou o aprendizado de máquina da seguinte forma (MITCHELL, 1997):

> "Um programa de computador aprende a partir de uma experiência E, em relação a alguma tarefa T e a uma determinada medida de performance P, se sua performance melhora a partir de E".

Para Holdsworth (2024), o aprendizado profundo (*deep learning*) é um subconjunto do aprendizado de máquina que usa redes neurais de várias camadas, chamadas de redes neurais profundas, com a finalidade de possibilitar o aprendizado de padrões complexos presentes nos dados, em geral, envolvendo múltiplos níveis de abstração. Cada camada da rede neural aprende a transformar a representação da camada anterior em uma representação um pouco mais abstrata e composta. Por exemplo, em um sistema de reconhecimento de imagem, a primeira camada pode aprender a detectar bordas, a segunda a compor essas bordas para detectar formas (círculos, quadrados), a terceira a compor as formas para detectar partes de um objeto (um olho, um nariz) e assim por diante. O aprendizado profundo avançou drasticamente o estado da arte em reconhecimento de fala, reconhecimento de objetos visuais, detecção de objetos e muitos outros problemas.

Para Goodfellow *et al.* (2016), o aprendizado profundo é uma forma de aprendizado de máquina que permite que os computadores aprendam com a experiência e entendam o mundo em termos de uma hierarquia de conceitos. Como o computador reúne conhecimento a partir da experiência, não há necessidade de um operador humano especificar formalmente todo o conhecimento de que o computador precisa. A hierarquia de conceitos permite que o computador aprenda conceitos complexos, construindo-os a partir de outros mais simples. Essa definição reforça a ideia de que o aprendizado profundo se destaca por sua capacidade de aprender características (*features*) automaticamente a partir dos dados, em uma estrutura hierárquica, eliminando a necessidade de engenharia de características manual, etapa crucial nos métodos de aprendizado de máquina tradicionais. Assim, enquanto os modelos de aprendizagem supervisionada exigem dados de entrada estruturados e rotulados para produzir resultados precisos, os modelos de aprendizado profundo podem usar aprendizagem não supervisionada (HOLDSWORTH, 2024).

As Redes Neurais Convolucionais (*CNN*s ou ConvNets) são uma classe especializada de redes neurais profundas, projetadas especificamente para processar dados que possuem uma topologia de grade, como as imagens, por exemplo. Uma imagem pode ser representada por uma grade de *pixels* (2D), e um vídeo é uma sequência de imagens (3D). As *CNN*s são arquitetadas para reconhecer padrões visuais diretamente a partir dos *pixels*, com o mínimo de pré-processamento. As redes neurais convolucionais são simplesmente redes neurais que usam a convolução no lugar da multiplicação de matrizes geral em pelo menos uma de suas camadas (GOODFELLOW *et al.*, 2016). Elas são inspiradas na organização do córtex visual de animais, projetadas para aprender de forma automática e adaptativa, em que neurônios individuais respondem a estímulos em regiões específicas do campo visual (MONTEIRO *et al.*, 2021). As *CNN*s são amplamente utilizadas em tarefas de visão computacional, como, por exemplo, a classificação de imagens, problema abordado neste trabalho. Uma grande vantagem das *CNN*s é que elas eliminam a necessidade de extração manual de características, realizando essa tarefa automaticamente.

A Figura 1 ilustra um exemplo de uma *CNN*. Mais especificamente, ela exibe a arquitetura de uma *CNN* denominada VGG-16 (LEARNOPENCV, 2024). Segundo (IBM, 2024) as *CNN*s estão estruturadas da seguinte maneira:

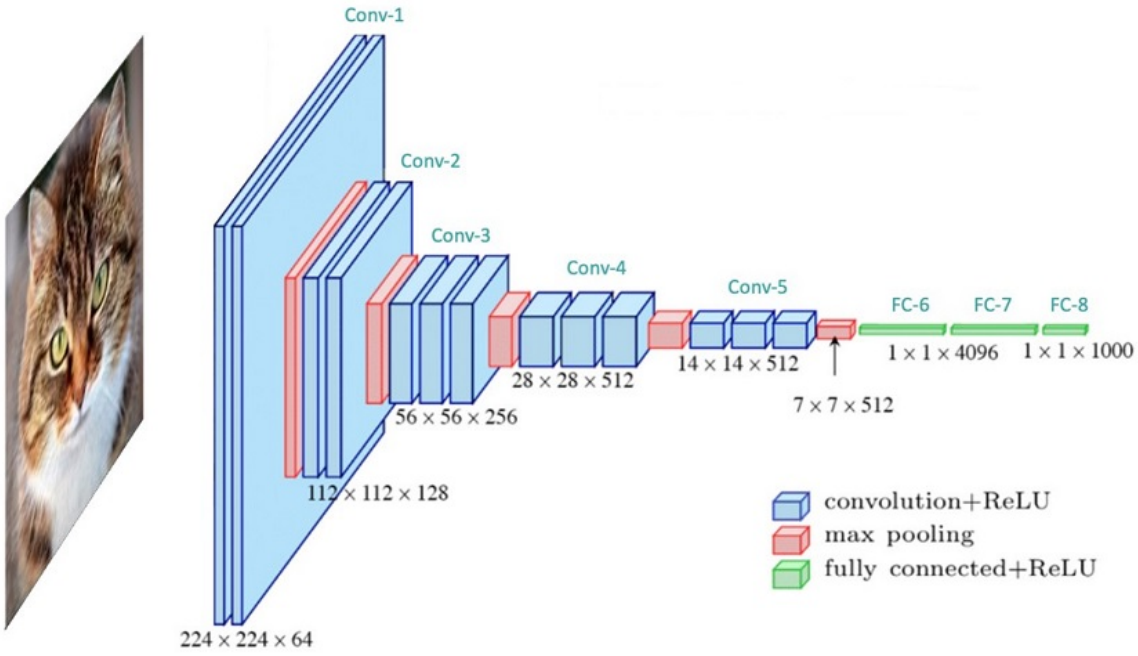


Figura 1 – Arquitetura da *CNN* VGG-16.
Fonte: (LEARNOPENCV, 2024)

Camada de Convolução: É a camada fundamental de uma *CNN*. Em vez de conectar cada neurônio da camada anterior a um determinado neurônio da camada seguinte (como em uma rede neural densa), a camada convolucional usa um filtro (ou *kernel*). Mais precisamente, um filtro é uma pequena matriz de pesos (por exemplo, 3x3 ou 5x5 *pixels*). Esse filtro desliza sobre toda a imagem de entrada, um trecho de cada vez, e realiza uma operação matemática (produto de ponto) entre os pesos do filtro e os valores dos *pixels* sob ele, resultando em um processo conhecido como convolução. O resultado dessa operação é um "mapa de características", que destaca a presença de um padrão específico (como uma borda vertical, uma curva ou uma cor) na imagem. A novidade aqui é o compartilhamento de parâmetros: O mesmo filtro é reutilizado em toda a imagem. Isso significa que, se a rede aprende a detectar uma borda vertical em um canto da imagem, ela pode detectar essa mesma borda em qualquer outra posição. Isso torna a rede invariante à translação e drasticamente reduz o número de parâmetros a serem aprendidos, tornando o treinamento mais eficiente. Em resumo, a camada de convolução é crucial para permitir que a rede aprenda e identifique recursos importantes que são essenciais para tarefas como classificação de imagens e detecção de objetos.

Camada de *Pooling*: Logo após uma camada convolucional, é comum usar uma camada de *pooling*. Sua função é reduzir a dimensionalidade espacial do mapa de características, o que ajuda a reduzir a complexidade computacional, controlar o sobreajuste (*overfitting*), bem

como criar uma certa invariância a pequenas translações e distorções na imagem. Assim como a camada convolucional, a operação de *pooling* utiliza um filtro que aplica uma função de agregação aos valores. O tipo mais comum de *pooling* é o *Max Pooling*, o qual seleciona o valor máximo de uma pequena janela (ex: 2x2 *pixels*) do mapa de características, descartando o restante. Apesar da perda de informações nesta etapa, ela possui a vantagem de reduzir a dimensionalidade dos mapas de características e consequentemente o número de parâmetros na entrada, mantendo as informações mais importantes e tornando a rede mais eficiente, além de limitar o risco de *overfitting*.

Camadas *Fully Connected*: Conecta cada neurônio da camada de entrada a um determinado neurônio da camada de saída, formando uma região densamente conectada. Essa camada executa a tarefa de classificação baseada nas características extraídas por meio das camadas anteriores e seus diferentes filtros. Normalmente utilizam uma função de ativação produzindo uma probabilidade de 0 a 1.

2.2 Aprendizagem por Transferência

Limitações práticas, como a quantidade restrita de dados de treinamento, dificultam que as redes neurais profundas, treinadas do zero, alcancem um desempenho satisfatório. Para superar essas limitações, uma alternativa é a Aprendizagem por Transferência (ou *Transfer Learning*), onde em vez de treinar um modelo do zero para uma nova tarefa, o que exige uma grande quantidade de dados e poder computacional, pegamos um modelo que já foi treinado em uma tarefa relacionada e o adaptamos para a nova tarefa. A ideia parte do comportamento humano, onde uma pessoa pode aplicar o conhecimento adquirido em uma determinada área (por exemplo, tocar piano) para aprender mais rapidamente uma nova área relacionada (tocar teclado). Com a aprendizagem por transferência, em vez de iniciar o processo de aprendizagem do zero, podemos começar a partir de valores previamente aprendidos ao solucionar um problema anterior (MONTEIRO *et al.*, 2021). A técnica se torna mais eficaz quando as tarefas relacionam-se a problemas semelhantes, como os enfrentados no de reconhecimento de imagem. Dessa forma, um modelo treinado para resolver uma tarefa específica é reutilizado como ponto de partida para abordar uma tarefa diferente, porém relacionada (MONTEIRO *et al.*, 2021).

O Pytorch permite o carregamento de pesos aprendidos através de treinamentos que utilizaram conjuntos de dados muito grandes, como, por exemplo, o ImageNet (CHILAMKURTHY, 2025). ImageNet é um vasto banco de imagens que contém mais de 14 milhões

de imagens anotadas, que é comumente utilizado na área de visão computacional para a classificação de imagens (IMAGENET, 2024). O uso dessa base permite que desenvolvedores treinem modelos de *CNN*s com uma grande quantidade de dados, possibilitando uma melhora na precisão dessas redes em diversas aplicações de classificação de imagem. Partindo desses valores que foram aprendidos, o novo processo de treinamento continua com as imagens do problema específico que se quer solucionar. Por exemplo, um modelo que foi treinado utilizando a base Imagenet pode ser usado como ponto de partida para o treino de um novo modelo a partir de uma base de dados com um menor número de imagens relacionadas a um problema específico.

De acordo com (BERKAYA *et al.*, 2021) a utilização do transferência de aprendizagem proporciona algumas vantagens, tais como:

- A redução do tempo de treinamento, pois o modelo já foi treinado em um grande conjunto de dados. Dessa forma, o tempo de convergência do modelo para um novo problema pode ser reduzido;
- Favorecer o desempenho com dados escassos, pois normalmente as *CNN*s precisam de uma grande quantidade de dados para produzir bons resultados, o que nem sempre é possível de se obter no problema investigado;
- Eficiência, pois utilizar um modelo pré-treinado pode economizar tempo e recursos computacionais acelerando a convergência da solução desenvolvida.

2.3 Quantização

Para facilitar a implantação de *CNN*s em sistemas com poder computacional restrito e/ou alimentados por baterias podemos utilizar técnicas de quantização.

A quantização é uma forma de compressão com perdas, onde a precisão da informação é reduzida em troca de uma representação mais compacta. Assim, quantização refere-se a técnicas para realizar cálculos e armazenar os seus resultados em uma estrutura de menor precisão.

A ideia de quantização tem sido aplicada com sucesso na área de aprendizado de máquina. A quantização de um modelo de aprendizado de máquina consiste no processo de converter os parâmetros (pesos) e/ou as ativações intermediárias de uma representação de ponto flutuante de alta precisão (geralmente 32 *bits*) para uma representação de ponto fixo ou inteiro de baixa precisão (como 8 *bits*, ou até menos). A quantização é uma técnica de otimização crucial, especialmente para implantar modelos de aprendizado de máquina em *hardware* com recursos

limitados

O *framework* PyTorch (KRISHNAMOORTHI *et al.*, 2024) fornece suporte para a utilização do conceito de quantização em modelos de aprendizado de máquina. Normalmente os modelos são representados em ponto flutuante de 32 *bits*. Todavia, com a quantização, eles podem ser representados em ponto flutuante de 16 *bits* ou em inteiros de 8 *bits*, por exemplo.

Em alguns casos a quantização permite uma redução de quatro vezes no tamanho do modelo e uma redução de quatro vezes nos requisitos de largura de banda (PYTORCH, 2024). Essa redução no espaço de armazenamento pode permitir um melhor uso da memória *cache* do sistema (WU *et al.*, 2020). Além disso, cálculos inteiros de 8 *bits* normalmente são duas a quatro vezes mais rápidos que os cálculos ponto flutuante de 32 *bits* (KRISHNAMOORTHI *et al.*, 2024). A complexidade computacional reduzida também pode resultar em um menor consumo de energia. Dessa forma, a quantização é uma técnica que pode acelerar o tempo de inferência, devido a redução de largura de banda e cálculos mais rápidos, além de diminuir o tamanho dos modelos o que possibilita que estes sejam armazenados e executados em equipamentos com recursos computacionais limitados e alimentados por baterias, tais como *smartphones*, sensores e dispositivos *Internet of Things* / Internet das Coisas (*IoT*).

Ao utilizar quantização, devemos considerar alguns fatores, tais como:

- Perda de acurácia: Quando o problema envolvido é complexo, a quantização pode resultar em perda de acurácia, introduzida pelas aproximações do processo de quantização (KRISHNAMOORTHI *et al.*, 2024) .
- Abordagem utilizada: Existem várias formas de quantização e elas podem ser realizadas em diferentes camadas e com variados parâmetros, sendo possível uma investigação dos melhores cenários.

Neste trabalho utilizamos a Quantização Dinâmica (ou *Dynamic Quantization*), também conhecida como "quantização por faixa dinâmica" (*dynamic range quantization*), é uma técnica de quantização pós-treinamento (*Post-Training Quantization*) em que os parâmetros do modelo (pesos) são convertidos para uma representação de menor precisão (ex.: *8-bit integer* / inteiro de 8 *bits* (*int8*)) antes da sua implantação. As ativações, no entanto, permanecem em ponto flutuante (*32-bit floating point number* / número de ponto flutuante de 32 *bits* (*float32*)) na memória e são quantizadas"*on-the-fly*" (dinamicamente) para valores de menor precisão (KRISHNAMOORTHI *et al.*, 2024).

3 REDES CONVOLUCIONAIS AVALIADAS

Neste capítulo, descreveremos brevemente cada uma das quatorze redes neurais convolucionais avaliadas neste trabalho.

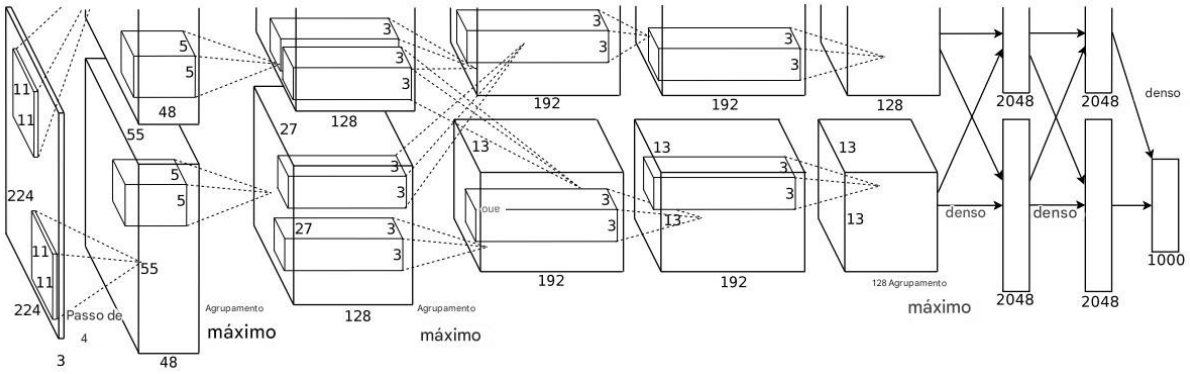
3.1 AlexNet

A rede convolucional AlexNet possui algumas características bem particulares e interessantes. Segundo (KRIZHEVSKY *et al.*, 2012) essas características são:

1. Não linearidade *ReLU*: AlexNet faz uso de unidades lineares retificadas (*ReLU*s) como função de ativação em vez de funções *tanh*. Essa escolha permite acelerar o treinamento do modelo, proporcionando forte influência no desempenho de grandes modelos treinados em grandes conjuntos de dados.
2. Treinamento em múltiplas *graphics processing unit* / unidade de processamento gráfico (*GPU*): A rede AlexNet permite o treinamento em duas *GPU*s, de forma paralela. Essas *GPU*s estão limitadas a se comunicarem apenas em algumas camadas. Esse esquema com duas *GPU*s resulta em um tempo de treinamento um pouco menor.
3. Normalização de resposta local: Essa normalização imita uma forma de inibição lateral inspirada nos neurônios biológicos reais, criando competição entre esses neurônios a partir de saídas calculadas utilizando diferentes filtros. Essa normalização reduz as taxas de erro.
4. *Pooling* sobreposto: As redes AlexNet utilizam *pooling* sobreposto. Essa abordagem ajuda a diminuir o sobreajuste e reduzir as taxas de erro.

A arquitetura da rede AlexNet é ilustrada na Figura 2. Ela contém oito camadas com pesos, as cinco primeiras são convolucionais e as três últimas são camadas completamente conectadas. Entre a primeira e a segunda camada convolucional, existe uma camada de normalização. O *ReLU* não linear é usado na saída de todas as camadas (convolucional e totalmente conectada) (KRIZHEVSKY *et al.*, 2012).

Figura 2 – Arquitetura da Rede Alexnet.

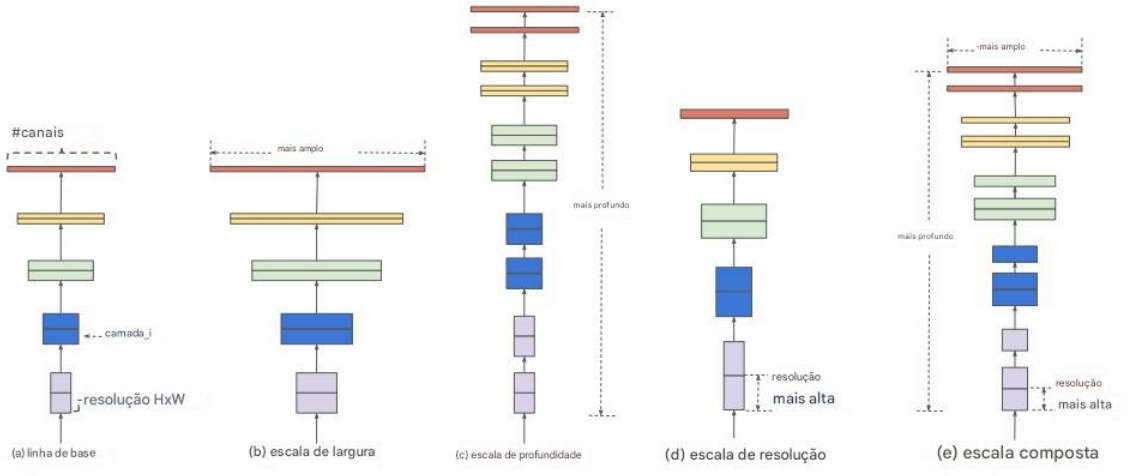


Fonte: (KRIZHEVSKY *et al.*, 2012) adaptado

3.2 EfficientNet

A rede EfficientNet adota um novo método de dimensionamento uniforme para as dimensões de profundidade, largura e resolução, usando um coeficiente composto, o que a diferencia das redes anteriores, que adotam um escalonamento arbitrário desses fatores (TAN; LE, 2019). Esse método de dimensionamento pode ser facilmente aplicado a outros modelos por meio de técnicas de transferência de aprendizado (TAN; LE, 2019). Na Figura 3 é possível visualizar o método de dimensionamento utilizado pela EfficientNet. Nela (a) é um exemplo de rede de referência; (b)-(d) são escalonamentos convencionais que aumentam apenas uma dimensão da rede, largura, profundidade ou resolução. (e) Método de escala composto proposto que escala uniformemente todas as três dimensões com uma proporção fixa (TAN; LE, 2019).

Figura 3 – Dimensionamento do modelo utilizado pela EfficientNet.



Fonte: (TAN; LE, 2019) adaptado

Tabela 1 – Rede base EfficientNet-B0 sendo seu principal bloco de construção é o *mobile inverted bottleneck* MBConv

Estágio	**Operador**	**Resolução (A x L)**	**Canais**	**Camadas**
1	Conv3x3	224 x 224	32	1
2	MBConv1, k3x3	112 x 112	16	1
3	MBConv6, k3x3	112 x 112	24	2
4	MBConv6, k5x5	56 x 56	40	2
5	MBConv6, k3x3	28 x 28	80	3
6	MBConv6, k5x5	14 x 14	112	3
7	MBConv6, k5x5	14 x 14	192	4
8	MBConv6, k3x3	7 x 7	320	1
9	Conv1x1 & Pooling & FC	7 x 7	1280	1

Fonte: (TAN; LE, 2019) adaptado

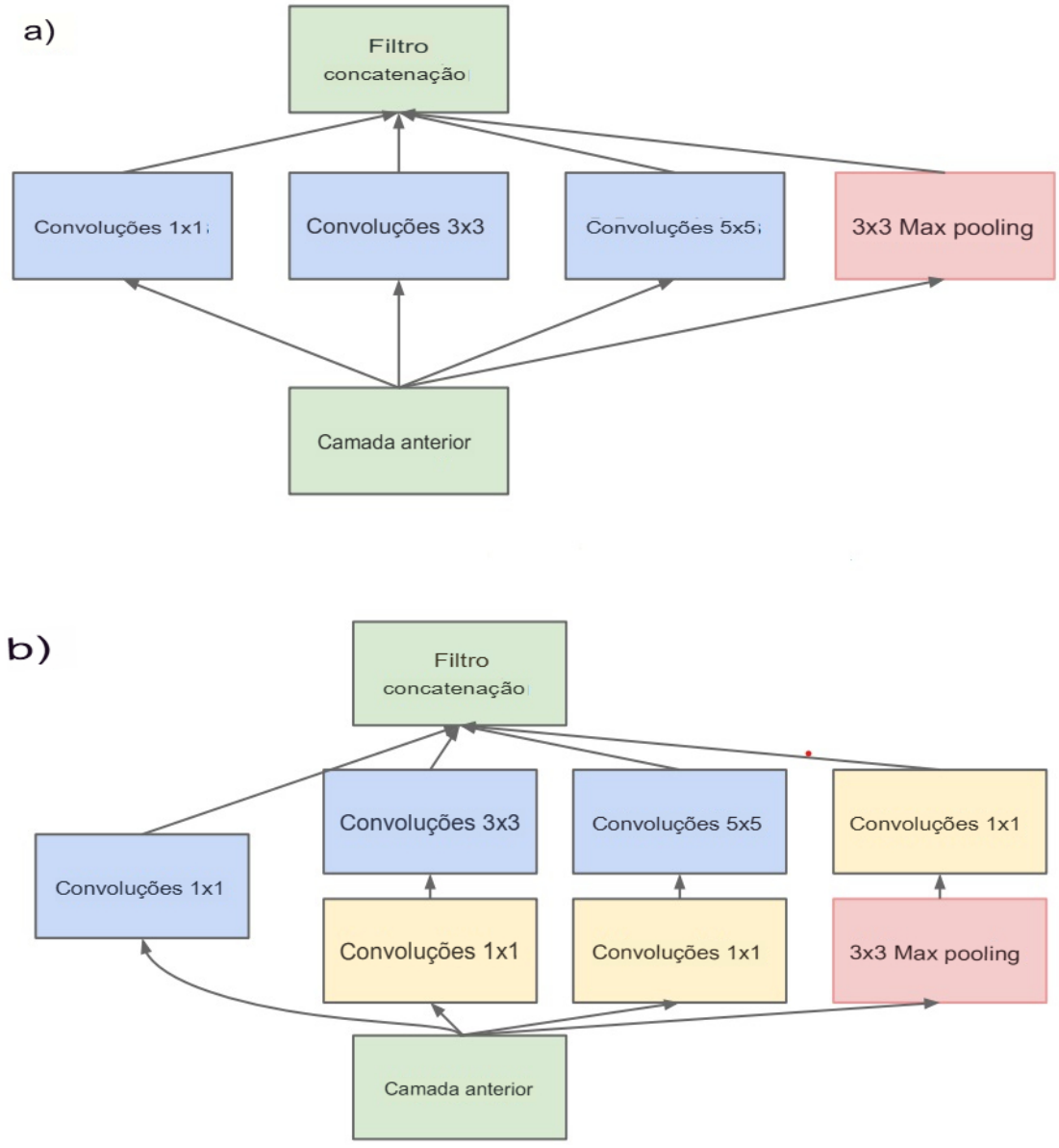
Os dimensionamentos são realizados por meio de coeficientes fixos determinados por meio de uma pesquisa de grade no modelo original (EfficientNet-B0), que serve de base para a criação dos outros modelos da família (do EfficientNet-B1 ao EfficientNet-B7). Dessa forma, a rede aumenta uniformemente o seu tamanho, a sua complexidade e a sua capacidade (TAN; LE, 2019). A rede EfficientNet-B0 é mostrada na Tabela 1.

3.3 GoogLeNet

O principal avanço proporcionado pela GoogLeNet está relacionado à melhor utilização dos recursos computacionais. Para isso, a rede possui maior profundidade e largura, ao mesmo tempo em que mantém um custo computacional uniforme (SZEGEDY *et al.*, 2015).

O modelo possui módulos *Inception*, que são os blocos de construção fundamentais. Esses módulos executam várias convoluções com diferentes tamanhos de filtro, sendo alguns filtros 1x1, além de filtros 3x3 e filtros 5x5, seguidos por uma operação de *pooling*. Cada módulo realiza uma concatenação das saídas das camadas convolucionais, como mostrado na Figura 4. A saída de um módulo serve como entrada para o próximo módulo. Essa abordagem faz com que a rede capture padrões em múltiplas escalas e melhore a eficiência computacional, além da capacidade de aprendizagem (SZEGEDY *et al.*, 2015).

Figura 4 – Módulo Inception (a) versão ingênua (b) com reduções de dimensões



Fonte: (SZEGEDY *et al.*, 2015) adaptado

Tabela 2 – Arquitetura GoogLeNet

Tipo	Tamanho de correção	Tamanho de saída	Profundidade	#1x1	#3x3	#5x5	#5x5	Pool	Params	Ops
Convolução	7x7/2	112x112x64	1	-	-	-	-	max pool	2.7K	34M
Max Pool	3x3/2	56x56x64	0	-	-	-	-	-	-	-
Convolução	3x3/1	56x56x192	2	64	192	-	-	max pool	112K	360M
Max Pool	3x3/2	28x28x192	0	-	-	-	-	-	-	-
Inception (3a)	-	28x28x256	2	64	96	128	16	32	32	159K
Inception (3b)	-	28x28x480	2	128	128	192	32	96	64	380K
Max Pool	3x3/2	14x14x480	0	-	-	-	-	-	-	-
Inception (4a)	-	14x14x512	2	192	96	208	16	48	64	364K
Inception (4b)	-	14x14x512	2	160	112	224	24	64	64	437K
Inception (4c)	-	14x14x512	2	128	128	256	24	64	64	463K
Inception (4d)	-	14x14x528	2	112	144	288	32	64	64	580K
Inception (4e)	-	14x14x832	2	256	160	320	32	128	128	840K
Max Pool	3x3/2	7x7x832	0	-	-	-	-	-	-	-
Inception (5a)	-	7x7x832	2	256	160	320	32	128	128	1072K
Inception (5b)	-	7x7x1024	2	384	192	384	48	128	128	1388K
Avg Pool	7x7/1	1x1x1024	0	-	-	-	-	-	-	-
Dropout (40%)	-	1x1x1024	0	-	-	-	-	-	-	-
Linear	-	1x1x1000	1	-	-	-	-	-	1000K	1M
Softmax	-	1x1x1000	0	-	-	-	-	-	-	-

Fonte: (SZEGEDY *et al.*, 2015) adaptado

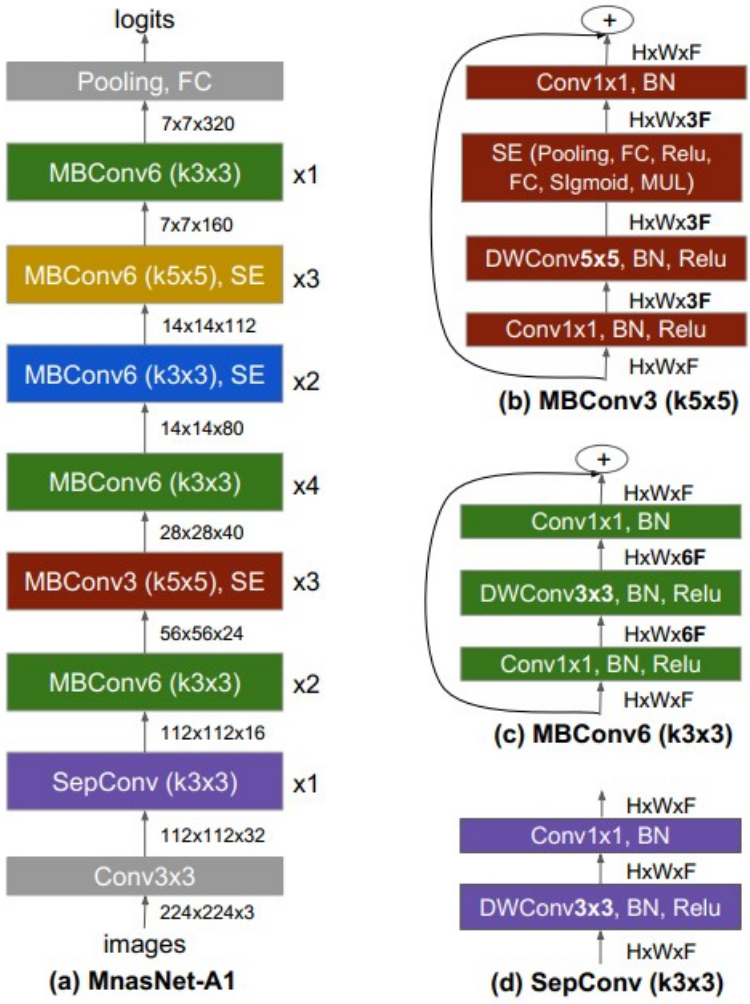
Na Tabela 2, é possível visualizar a arquitetura do GoogLeNet. Nessa arquitetura, todas as convoluções, inclusive as dentro dos módulos *Inception*, utilizam ativação linear retificada. O tamanho da entrada é 224×224, considerando os canais de cores *red, green, blue* (*RGB*). Na Tabela 2, os campos "*3×3 reduce*" e "*5×5 reduce*" representam o número de filtros 1×1 na redução de camadas usadas antes das convoluções 3×3 e 5×5. Pode-se ver o número de filtros 1×1 na camada de projeção, após o *pool* máximo integrado, na coluna "*pool proj*". Todas essas reduções/projeções de camadas também usam ativação linear retificada (SZEGEDY *et al.*, 2015).

3.4 MnasNet

A rede MnasNet foi construída com o objetivo de possibilitar uma maior eficiência em dispositivos móveis. Esses dispositivos requerem uma arquitetura que ofereça um certo equilíbrio entre precisão e latência, o que, em geral, era alcançado de forma manual, demorando bastante tempo. Para solucionar este problema, a MnasNet utiliza um método automático de busca de arquiteturas neurais, otimizado e adaptado para dispositivos móveis. O método de busca (de arquiteturas neurais) explora um espaço de busca hierárquico, incentivando a diversidade de camadas e reduzindo o tamanho do espaço de busca. Para isso, o método baseia-se em aprendizado por reforço (TAN *et al.*, 2019). A rede MnasNet foi avaliada diretamente em dispositivos móveis, buscando uma maior aproximação com aplicações reais (TAN *et al.*, 2019).

Na Figura 5 podemos visualizar a arquitetura da rede MnasNet. Observe que a arquitetura da rede MnasNet utiliza tanto convoluções 3x3 quanto 5x5, diferentemente dos modelos voltados para dispositivos móveis, que utilizam apenas convoluções 3x3. MBConv denota convolução de gargalo invertido móvel, DWConv denota convolução em profundidade, k3x3/k5x5 indica o tamanho do *kernel*, BN é a norma do lote, HxWxF denota a forma do tensor (altura, largura,profundidade) e ×1=2=3=4 denota o número de camadas repetidas dentro do bloco (TAN *et al.*, 2019).

Figura 5 – Arquitetura MnasNet (a); do (b) ao (d) são algumas estruturas de camadas correspondentes.



Fonte: (TAN *et al.*, 2019) adaptado

3.5 MobileNetV2

MobileNetV2 é uma arquitetura de rede convolucional desenvolvida tanto para aplicações *web* quanto para aplicações móveis. Segundo (SANDLER *et al.*, 2018) as principais características da rede MobileNetV2 são:

- Separação das camadas de convolução é uma estratégia que quebra a operação de convolução em outras duas operações, uma operação de convolução em profundidade, através da aplicação de um filtro por canal de entrada, e uma operação de convolução pontual, que realiza uma convolução 1x1 na combinação das saídas da convolução em profundidade.
- A técnica de resíduos invertidos inverte o processo de expansão do espaço de atributos. Tradicionalmente, os modelos expandem as representações na entrada de um bloco residual e depois diminuem. O MobileNetV2 começa usando camadas de gargalo que reduzem o número de canais antes de aplicar a conexão residual e aumentar o número de canais até o tamanho original. Essa estratégia torna mais eficiente a extração de características.
- Outra técnica utilizada é a de gargalos lineares na estrutura residual invertida, cuja finalidade consiste em remover as não linearidades (*ReLU*) dos gargalos do modelo. Esta remoção ajuda a preservar as informações e a manter o poder representacional da rede, evitando transformações desnecessárias.

As técnicas utilizadas na MobileNetV2 resultam na redução do custo computacional e do tamanho do modelo, bem como no uso mais eficiente dos parâmetros e da memória, facilitando sua utilização em aplicações móveis e embarcadas. A arquitetura da rede MobileNetV2 pode ser vista na Tabela 3 em que cada linha descreve uma sequência de uma ou mais camadas idênticas (módulo *stride*), repetidas "n"vezes. Todas as camadas na mesma sequência têm o mesmo número "c"de canais de saída. A primeira camada de cada sequência tem um passo "s"e todas as outras usam passo 1. Todas as convoluções espaciais usam filtros 3×3 (SANDLER *et al.*, 2018).

Tabela 3 – Arquitetura MobileNetV2.

Entrada	Operador	t	c	n	s
224^2 x 3	conv2d	-	32	1	2
112^2 x 32	bottleneck	1	16	1	1
112^2 x 16	bottleneck	6	24	2	2
56^2 x 24	bottleneck	6	32	3	2
28^2 x 32	bottleneck	6	64	4	2
14^2 x 64	bottleneck	6	96	3	1
14^2 x 96	bottleneck	6	160	3	2
7^2 x 160	bottleneck	6	320	1	1
7^2 x 320	conv2d 1x1	-	1280	1	1
7^2 x 1280	avgpool 7x7	-	-	1	-
1 x 1 x 1280	conv2d 1x1	-	k	-	-

Fonte: (SANDLER *et al.*, 2018) adaptado

3.6 MobileNetV3

MobileNetV3 é uma rede convolucional voltada para *central processing unit* / unidade central de processamento (*CPU*) de telefones celulares. Ela tem como objetivo alcançar baixa latência e baixo consumo de energia, sendo, assim, adequada para aplicações em dispositivos móveis.

A arquitetura da rede MobileNetV3 explora uma combinação de técnicas, em que cada técnica funciona com um bloco de construção. Assim, cada camada é formada por um conjunto de blocos de construção, ou seja, por uma combinação de técnicas. De acordo com (HOWARD *et al.*, 2019) as principais técnicas utilizadas pela rede MobileNetV3 são:

- **Convoluções separáveis em profundidade**: convoluções separáveis em profundidade são mais eficientes do que as camadas de convolução tradicionais. Elas são formadas por duas camadas separadas, uma de convolução profunda leve, para filtragem espacial, e outra camada de convolução pontual 1x1, mais pesada, a qual é utilizada para geração de atributos. Esta técnica foi utilizada no modelo MobileNetV1 .
- **Gargalo linear e estrutura residual invertida**: essas técnicas permitem a criação de estruturas de camadas ainda mais eficientes. Elas foram utilizadas no modelo MobileNetV2;
- **Módulos de atenção leve**: esses módulos são baseados em *squeeze and excitation*, sendo utilizados na estrutura do gargalo.

Tabela 4 – Especificação do modelo MobileNetV3-Large.

Entrada	Operador	exp size	#out	SE	NL	s
224^2 x 3	conv2d	-	16	-	HS	2
112^2 x 16	bneck, 3x3	16	16	-	RE	1
112^2 x 16	bneck, 3x3	64	24	-	RE	2
56^2 x 24	bneck, 3x3	72	24	-	RE	1
56^2 x 24	bneck, 5x5	72	40	✓	RE	2
28^2 x 40	bneck, 5x5	120	40	✓	RE	1
28^2 x 40	bneck, 5x5	120	40	✓	RE	1
28^2 x 40	bneck, 3x3	240	80	-	HS	2
14^2 x 80	bneck, 3x3	200	80	-	HS	1
14^2 x 80	bneck, 3x3	184	80	-	HS	1
14^2 x 80	bneck, 3x3	184	80	-	HS	1
14^2 x 80	bneck, 3x3	480	112	✓	HS	1
14^2 x 112	bneck, 3x3	672	112	✓	HS	1
14^2 x 112	bneck, 5x5	672	160	✓	HS	2
7^2 x 160	bneck, 5x5	960	160	✓	HS	1
7^2 x 160	bneck, 5x5	960	160	✓	HS	1
7^2 x 160	conv2d, 1x1	-	960	-	HS	1
7^2 x 960	pool, 7x7	-	-	-	-	-
1^2 x 960	conv2d 1x1, NBN	-	1280	-	HS	1
1^2 x 1280	conv2d 1x1, NBN	-	k	-	-	-

Fonte: (HOWARD *et al.*, 2019) adaptado

Finalmente, duas versões de rede MobileNetV3 foram propostas: MobileNetV3-Large para aparelhos com maiores recursos computacionais e MobileNetV3-Small para aparelhos com menores recursos.

A Tabela 4 apresenta a especificação do modelo MobileNetV3-Large. Nela "SE"denota se há um *Squeeze-And-Excite* nesse bloco. “NL” denota o tipo de não linearidade usada. “HS” denota *h-swish* e “RE” denota *ReLU*. “NBN” denota nenhuma normalização de lote e o “s” denota o passo.

Tabela 5 – Especificação do modelo MobileNetV3-Small.

Entrada	Operador	exp size	#out	SE	NL	s
224^2 x 3	conv2d, 3x3	-	16	-	HS	2
112^2 x 16	bneck, 3x3	16	16	✓	RE	2
56^2 x 16	bneck, 3x3	72	24	-	RE	2
28^2 x 24	bneck, 3x3	88	24	-	RE	1
28^2 x 24	bneck, 5x5	96	40	✓	HS	2
14^2 x 40	bneck, 5x5	240	40	✓	HS	1
14^2 x 40	bneck, 5x5	240	40	✓	HS	1
14^2 x 40	bneck, 5x5	120	48	✓	HS	1
14^2 x 48	bneck, 5x5	144	48	✓	HS	1
14^2 x 48	bneck, 5x5	288	96	✓	HS	2
7^2 x 96	bneck, 5x5	576	96	✓	HS	1
7^2 x 96	bneck, 5x5	576	96	✓	HS	1
7^2 x 96	conv2d, 1x1	-	576	✓	HS	1
7^2 x 576	pool, 7x7	-	-	-	-	-
1^2 x 576	conv2d 1x1, NBN	-	1024	-	HS	1
1^2 x 1024	conv2d 1x1, NBN	-	k	-	-	-

Fonte: (HOWARD *et al.*, 2019) adaptado

A Tabela 5 contém a especificação do modelo MobileNetV3-Small. As colunas "SE", "NL", "HS" ,"RE","NBN" e "s" possuem o mesmo significado da Tabela 4.

3.7 RegNet

A RegNet (*Regular Network*) é uma arquitetura de rede neural convolucional (*CNN*) desenvolvida pela *Facebook Artificial Intelligence Research* (*FAIR*) . A principal inovação da RegNet não reside na criação de um único modelo de rede, mas sim na introdução de um novo paradigma para o projeto de arquiteturas de redes neurais. Em vez de treinar instâncias de redes individualmente, a abordagem da RegNet foca em projetar espaços de projeto de redes (*network design spaces*) que parametrizam populações inteiras de modelos. Essa estratégia permite a descoberta de princípios de projeto mais gerais e robustos. O resultado é um espaço de projeto de baixa dimensão, composto por redes simples e regulares, que oferece modelos rápidos e eficientes. Notavelmente, as redes RegNet superam em desempenho a popular arquitetura EfficientNet, sendo até 5 vezes mais rápidas (RADOSAVOVIC *et al.*, 2020).

A metodologia central por trás da RegNet baseia-se na ideia de espaços de projeto. Um espaço de projeto é um conjunto parametrizado de possíveis arquiteturas de *CNN*s. A qualidade de um espaço de projeto é avaliada por meio da amostragem de modelos, seguida do

treinamento e da análise estatística de suas distribuições de erro. Este processo permite identificar princípios de projeto que melhoram a população de modelos como um todo, em vez de otimizar uma única instância para um cenário específico (RADOSAVOVIC *et al.*, 2020).

O processo inicia-se com um espaço de projeto relativamente irrestrito, chamado *AnyNet*, no qual parâmetros como largura e profundidade da rede podem variar livremente ao longo dos estágios. Através de um processo iterativo de refinamento, guiado por ferramentas estatísticas como a Função de Distribuição Empírica de erros, o espaço de projeto é progressivamente simplificado. O objetivo é chegar a um espaço de projeto refinado que apresente maior concentração de modelos de alto desempenho (RADOSAVOVIC *et al.*, 2020).

O *insight* fundamental do espaço de projeto RegNet é que as larguras e profundidades das redes de bom desempenho podem ser descritas por uma função linear quantizada. Isso resulta em uma parametrização simples e em um espaço de projeto de baixa dimensão, mais fácil de analisar e interpretar.

O projeto básico das redes *AnyNet* é composto por um tronco, seguido pelo corpo da rede, que executa a maior parte da computação, e por uma rede final que prevê as classes de saída, como pode ser visto na Figura 6a. A maioria das alterações ocorre no corpo da rede (Figura 6b) que é formado por quatro estágios operando com resolução progressivamente reduzida. Cada etapa consiste em sequências de blocos idênticos, como pode ser visto na Figura 6c. Para cada estágio i, os graus de liberdade incluem o número de blocos, largura do bloco, razão de gargalo e largura do grupo. A maior parte dos testes realizados pelos autores usa o bloco de gargalos residuais padrão mostrado na Figura 7, o qual foi nomeado como bloco X (RADOSAVOVIC *et al.*, 2020).

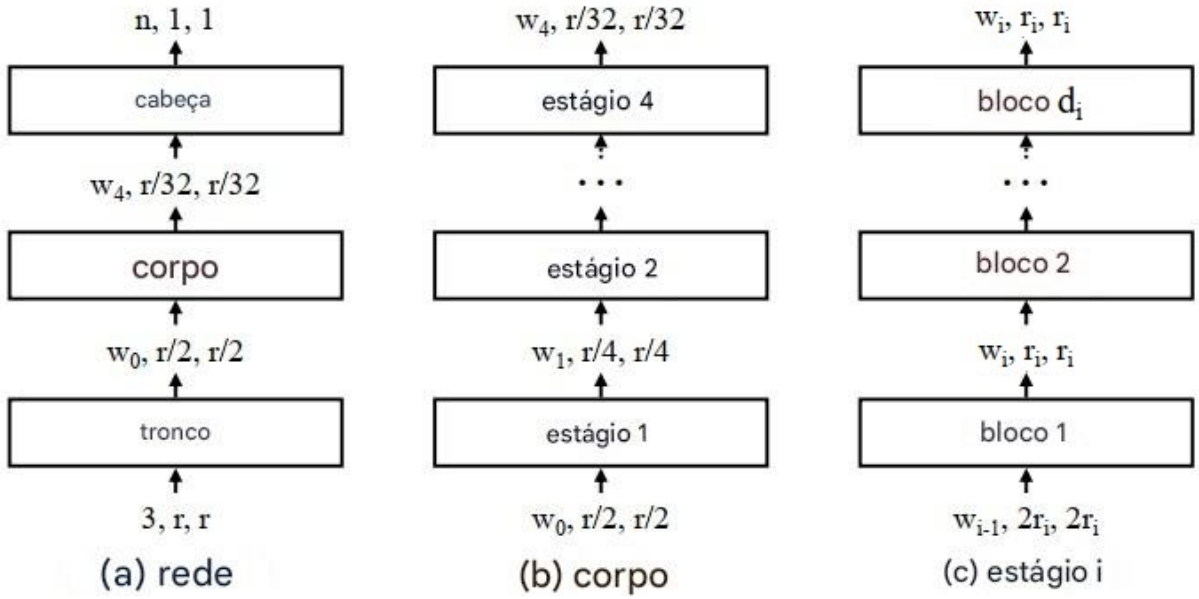
Mais especificamente, a RegNet combina o projeto manual e o *neural architecture search* / pesquisa de arquiteturas de rede (*NAS*). O projeto manual tem como vantagens a interpretabilidade e a descoberta de princípios gerais que formam redes simples. Por outro lado, o *NAS* ajuda o processo de busca ao tornar alguns procedimentos semiautomáticos (RADOSAVOVIC *et al.*, 2020).

A Figura 6 representa a estrutura geral de rede para modelos do projeto de espaços. Sendo que de acordo com (RADOSAVOVIC *et al.*, 2020):

- (a) cada rede consiste em um tronco (*stride-two* 3×3 conv com w0 = 32 canais de saída), seguido do corpo da rede que executa a maior parte do cálculo e, em seguida, uma cabeça (média *pooling* seguida por uma camada totalmente conectada) que prevê "n" saídas.
- (b) o corpo da rede é composto por uma sequência de estágios que operam com resolução progressivamente reduzida.
- (c) Cada etapa consiste em uma sequência de blocos idênticos, exceto o primeiro bloco que usa "conv" passo dois.

Embora a estrutura geral seja simples, o número total de configurações de rede possíveis é vasto .

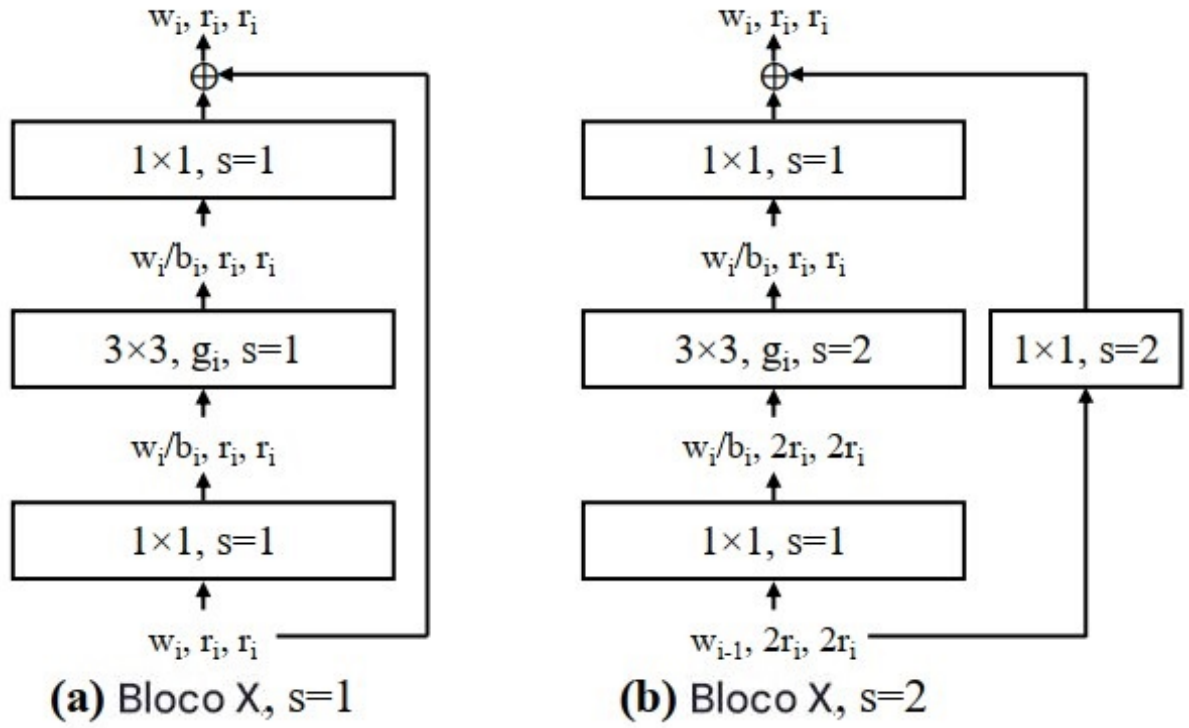
Figura 6 – Estrutura geral de rede para modelos do projeto de espaços.



Fonte: (RADOSAVOVIC *et al.*, 2020) adaptado

Na Figura 7 o bloco X é baseado no gargalo residual padrão com convolução de grupo. (a) Cada bloco X consiste em uma "Conv. 1×1", uma "conv." de grupo 3×3 e uma "conv" final 1×1, onde o 1×1 "convs" alteram a largura do canal. *BatchNorm* e *ReLU* seguem cada "conv". O bloco possui 3 parâmetros: largura wi, gargalo de proporção bi e largura do grupo gi. (b) A versão passo dois (s = 2) (RADOSAVOVIC *et al.*, 2020).

Figura 7 – O bloco de gargalos residuais (bloco X).



Fonte: (RADOSAVOVIC *et al.*, 2020) adaptado

Segundo (RADOSAVOVIC *et al.*, 2020) a partir do espaço de projeto RegNet, duas famílias principais de modelos foram propostas: RegNetX e RegNetY.

1. RegNetX: A família RegNetX representa os modelos base gerados pelo espaço de design. Eles são compostos por blocos residuais padrão, semelhantes aos usados na arquitetura ResNeXt, que utilizam convoluções em grupo. São modelos simples, eficientes e que servem como uma base forte para diversas tarefas.
2. RegNetY: A família RegNetY é uma evolução da RegNetX, que incorpora mecanismos de atenção de canal. A principal diferença é a adição de um módulo *Squeeze-and-Excitation* após cada camada convolucional de 3x3 nos blocos residuais. O módulo *Squeeze-and-Excitation* permite que a rede pondere dinamicamente a importância de cada canal de atributo, melhorando a capacidade de representação do modelo com um custo computacional adicional mínimo. Isso geralmente resulta em uma precisão superior em comparação com os modelos RegNetX correspondentes.

As famílias de modelos também foram formadas com um número crescente de parâmetros e de complexidade computacional, em que o sufixo do modelo indica essa complexidade. Dessa forma, o modelo RegNetX-200MF possui 200 milhões de operações de ponto flutuante, enquanto o RegNetX-400MF possui 400 milhões.

3.8 Visual Geometry Group (VGG)

A arquitetura de rede neural convolucional *Visual Geometry Group* (*VGG*) marcou um ponto de virada na área de visão computacional. Desenvolvida por Karen Simonyan e Andrew Zisserman na Universidade de Oxford, a *VGG* foi apresentada na competição *ImageNet Large Scale Visual Recognition Challenge* de 2014. Embora tenha ficado em segundo lugar na tarefa de classificação, atrás da GoogLeNet, a sua simplicidade, uniformidade e, principalmente, a sua profundidade, estabeleceram novos padrões para o projeto de redes neurais e influenciaram profundamente as arquiteturas subsequentes (SIMONYAN, 2014).

A principal contribuição da *VGG* foi demonstrar empiricamente que a profundidade da rede é um fator crítico para o desempenho em tarefas de reconhecimento de imagem. Ao aumentar o número de camadas para 16 e 19, a *VGG* superou significativamente as arquiteturas anteriores, que eram consideravelmente mais rasas (SIMONYAN, 2014).

A *VGG* adota uma abordagem uniforme, baseada em dois princípios centrais:

1. Filtros de Convolução Pequenos (3x3): A *VGG* utiliza exclusivamente filtros de convolução de 3x3, o menor tamanho capaz de capturar as noções de esquerda/direita, cima/baixo e centro. A inovação está em empilhar múltiplas camadas com esses filtros pequenos para simular o campo receptivo de filtros maiores. Por exemplo, duas camadas de convolução 3x3 empilhadas têm um campo de entrada efetivo de 5x5, enquanto três camadas equivalem a um campo de 7x7. Essa abordagem traz duas vantagens principais:
    - Aumento da Não-Linearidade: Empilhar múltiplas camadas convolucionais implica empregar múltiplas funções de ativação *ReLU*. Isso torna a função de decisão mais discriminativa.
    - Redução de Parâmetros: Uma pilha de filtros 3x3 tem menos parâmetros do que uma única camada com um filtro maior, mas com o mesmo campo receptivo.

Tabela 6 – Configurações ConvNet (mostradas em colunas). A função de ativação *ReLU* não é mostrada por questões de brevidade.

Configuração ConvNet					
A	A-LRN	B	C	D	E
11 camadas de peso	11 camadas de peso	13 camadas de peso	16 camadas de peso	16 camadas de peso	19 camadas de peso
entrada (224 x 224 imagem RGB)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-642
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-1283
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max4					

Fonte: (SIMONYAN, 2014) adaptado

A arquitetura *VGG* é composta por uma sequência de blocos convolucionais, seguida por camadas totalmente conectadas. Cada bloco consiste em algumas camadas de convolução 3x3, seguidas por uma camada de *max-pooling* 2x2 com *stride* 2, que reduz a resolução espacial pela metade. O número de filtros convolucionais começa em 64 no primeiro bloco e dobra a cada bloco subsequente, até atingir 512.

A Tabela 6 ilustra diferentes configurações da *VGG* (A, A-LRN, B, C, D, E), que se diferenciam pela profundidade e pela presença de "Normalização de Resposta Local" (LRN). A profundidade das configurações aumenta da esquerda (A) para a direita (E), à medida que mais camadas são adicionadas (as camadas adicionadas estão em negrito). Os parâmetros da camada convolucional são denotados como "conv⟨tamanho do campo receptivo⟩-⟨número de canais> " (SIMONYAN, 2014).

4 TRABALHOS RELACIONADOS

Stojnić *et al.* (2018) analisaram alguns métodos de detecção de abelhas portadoras de pólen em imagens obtidas na entrada da colmeia. A abordagem proposta é dividida em duas partes. Na primeira parte usaram dois métodos de segmentação baseados em descritores de cores para separar as abelhas. Na segunda parte usaram esses segmentos para classificar as abelhas em duas classes com ou sem pólen. A classificação é realizada por um modelo obtido a partir da utilização do método *Support Vector Machine* / Máquina de Vetores de Suporte (*SVM*), o qual foi treinado com poucas variações de descritores *VLA-Dencoded* e *SIFT*. O modelo obtido foi avaliado, utilizando a curva *Receiver Operating Characteristic* (*ROC*) como métrica de desempenho, obtendo uma pontuação de 0,9150 na classificação de um conjunto de 1000 imagens.

Babic *et al.* (2016) propõem um sistema não invasivo, baseado em sistemas embarcados, para detectar abelhas a partir de vídeos capturados na entrada da colmeia. Os autores aplicaram a Subtração de fundo, segmentação de cores e morfologia para segmentação de abelhas. Posteriormente, separaram as imagens segmentadas em duas classes: abelhas com pólen e sem pólen. Para a tarefa de classificação foi utilizado o algoritmo *K-Nearest Neighbors* / K-vizinhos mais próximos (*KNN*) com um descritor simples, que consiste em variações de cores e recursos de excentricidade, treinado a partir de um conjunto formado por 100 imagens. O classificador obtido apresentou uma taxa de classificação correta de 0,887.

Rodriguez *et al.* (2018) usaram um sistema embarcado para filmar as abelhas entrando na colmeia. Posteriormente, essas gravações foram recortadas e transformadas em imagens categorizadas em duas classes: abelhas com e sem pólen. Na tarefa de classificação foram usados três algoritmos clássicos de aprendizado de máquina: *KNN*, *Naive Bayes statistical*, e *SVM* com funções de *kernel* linear e não-linear. Além disso, foram avaliadas duas arquiteturas de *CNN* rasas, com uma e duas camadas, e três arquiteturas profundas (VGG16, VGG19 e ResNet50). A comparação entre os classificadores avaliados foi realizada utilizando-se de três variações de cores. Os melhores resultados, em termos de acurácia, foram obtidos pelos classificadores *SVM* com *Principal Component Analysis* (*PCA*), que alcançou uma acurácia de 0,9116, *CNN* rasa, que obteve uma acurácia de 0,964 e pela VGG19, a qual alcançou uma acurácia de 0,902.

Sledevič (2018) avaliou a classificação de imagens de abelhas com ou sem pólen utilizando uma *CNN*s de 1 a 3 camadas ocultas, com até 15 filtros por camada e tamanhos de filtro variando de 15x15 até 3x3 com o objetivo de criar modelos de baixo custo. A *CNN* de três camadas ocultas obteve acurácia de 0,94.

Yang e Collins (2019) usaram uma *CNN* profunda para medir e detectar sacos de pólen a partir de vídeos de monitoramento de abelhas. Eles utilizaram uma arquitetura *Faster Region-based Convolutional Neural Network* (*faster RCNN*) com rede central VGG16. A rede também consegue detectar se uma abelha é portadora de pólen ou não, permitindo sua contagem. A arquitetura utilizada apresentou uma taxa de acurácia de aproximadamente 0,96.

Ngo *et al.* (2021) usaram o YOLOv3-tiny, um modelo de detecção de objetos e classificação baseado em aprendizado profundo, apoiado por um algoritmo de rastreamento de objetos. Esse modelo foi treinado para reconhecer e contar abelhas, além de classificá-las como portadoras de pólen ou não. O modelo utilizado obteve um *F1-score* de 0,94, precisão de 0,91 e revocação de 0,99 para o problema do reconhecimento de abelhas portadoras ou não de pólen.

Berkaya *et al.* (2021) exploraram três bases de dados distintas, sendo uma delas voltada à classificação de abelhas como portadoras ou não de pólen. Os autores avaliaram os seguintes modelos de *Deep Convolutional Neural Networks* / Redes Neurais Convolucionais Profundas (*DNN*): AlexNet, DenseNet-201, GoogLeNet, ResNet-101, ResNet-18, VGG-16 e VGG-19. Além disso, para cada modelo, eles utilizaram os seguintes métodos: *Transfer Learning*, *SVM* com *deep features*[1], *SVM* com *shallow features*[2] e *SVM* com *deep and shallow features*. O melhor resultado foi obtido pelo GoogLeNet com *transfer learning*, o qual obteve uma acurácia de 0,9907, revocação de 1,0 e *F1-score* de 0,9905. AlexNet e VGG-16, ambas usando *transfer learning*, resultaram em uma precisão de 1,0.

Em Monteiro *et al.* (2021), os autores avaliaram a utilização de algumas *CNN*s na tarefa de detecção de abelhas portadoras de pólen. As *CNN*s avaliadas foram: VGG16, VGG19, ResNet50, ResNet101, InceptionV3, Inception-ResNetV2, Xception, DenseNet201 e DarkNet53.

Eles também testaram diferentes estratégias de pré-processamentos de imagens, incluindo a utilização das imagens originais, imagens em tons de cinza, além de técnicas de *Contrast Limit Adaptive Histogram Equalization* e *Unsharp Masking*. Os melhores resultados foram obtidos com o pré-processamento de imagens usando *Unsharp Masking* nos modelos

[1] *SVM* com características extraídas por redes neurais profundas, tais como *VGG* e ResNet.

[2] *SVM* com características extraídas por métodos tradicionais de visão computacional ou aprendizado de máquina, tais como: HOG, SIFT, LBP e SURF.

DarkNet53, resultando em uma acurácia de 0,991, e VGG16, com acurácia de 0,986.

Em Nguyen *et al.* (2024), os autores propuseram um conjunto de dados anotado para a detecção e classificação de abelhas portadoras de pólen. Este conjunto de dados contém 60.826 caixas anotadas, as quais delineiam abelhas portadoras de pólen e não portadoras de pólen, em 2.051 imagens capturadas nas entradas de colmeias. Adicionalmente, os autores utilizam dois modelos de detecção de objetos (YOLOv5 e *faster RCNN*) como base para incorporação de técnicas para lidar com o tamanho reduzido dos sacos de pólen e com o desbalanceamento das imagens, uma vez que o número de abelhas portadoras de pólen é menor que o de não portadoras. Os resultados mostraram que a abordagem proposta supera os modelos anteriores, no conjunto de dados proposto, com precisão, revocação e *F1-score* de 0,99, 0,93, e 0,95, respectivamente.

A Tabela 7 apresenta a principal diferença desse trabalho em relação aos trabalhos relacionados. Observe que o presente trabalho, diferentemente dos trabalhos anteriores, concentra-se em avaliar o uso da técnica de quantização em diferentes *CNN*s na solução do problema de classificação de abelhas, com e sem pólen, a partir de imagens coletadas nas entradas de colmeias.

Tabela 7 – Análise comparativa dos trabalhos relacionados

Trabalho	Quantização	Métodos usados	Base de dados	Resultados
Babic et al., 2016	Não	KNN	100 imagens.	Classificação correta de 0,887
Stojnić et al., 2018	Não	SVM, SIFT, VLAD	1000 imagens com resolução de 86x86 pixels.	Área sob a curva (AUC) de 0,9150
Rodriguez et al., 2018	Não	KNN, Naive Bayes statistical, SVM, CNN (1 e 2 camadas ocultas), VGG16, VGG19 e ResNet50	710 imagens com resolução de 180x300 pixels.	CNN rasa com acurácia de 0,964
Sledević, 2018	Não	CNN (1 a 3 camadas ocultas)	2000 imagens com resolução de 100x100 pixels.	Acurácia de 0,94 na CNN de três camadas ocultas
Yang and Collins, 2019	Não	Faster RCNN	2400 imagens.	Acurácia de aproximadamente 0,96
Ngo et al., 2021	Não	YOLOv3-tiny	3500 imagens com resolução de 640x480 pixels.	F1-score de 0,94, precisão de 0,91 e revocação de 0,99
Berkaya et al., 2021	Não	AlexNet, DenseNet-201, GoogLeNet, ResNet-101, ResNet-18, VGG-16 e VGG-19	714 imagens com resolução de 180x300 pixels.	Acurácia de 0,9907, revocação de 1,0 e F1-score de 0,9905
Monteiro et al., 2021	Não	VGG16, VGG19, ResNet50, ResNet101, InceptionV3, Inception-ResNetV2, Xception, DenseNet201 e DarkNet53	714 imagens com resolução de 180x300 pixels.	Acurácia de 0,991
Nguyen et al., 2024	Não	YOLOv5 e Faster RCNN	60.826 caixas anotadas.	Precisão, revocação e F1-score de 0,99, 0,93, e 0,95, respectivamente.
Este trabalho	Sim	Alexnet, EfficientNet-B0, GoogLeNet, MnasNet-0_5, MnasNet-1_3, MobileNetV2, MobileNetV3-Small, MobileNetV3-Large, RegNet-X400MF, RegNet-Y1_6GF, RegNet-Y400MF, RegNet-Y800MF e VGG16	715 imagens com resolução de 180x300 pixels.	Melhorara no tempo de inferência e/ou o tamanho de modelos.

5 METODOLOGIA EXPERIMENTAL

5.1 Classificadores Utilizados

Este trabalho avalia quatorze modelos de redes neurais convolucionais profundas para a classificação de abelhas que transportam e não transportam pólen, a partir de imagens obtidas na entrada da colmeia. A maioria desses modelos ainda não havia sido explorada na solução do problema da classificação de imagens de abelhas com ou sem pólen, incluindo os modelos: EfficientNet-B0, EfficientNet-B1, MnasNet0_5, MnasNet1_3, MobileNetV2, RegNet-X400MF, RegNet-Y1_6GF, RegNet-Y400MF, RegNet-Y800MF. Adicionalmente, outros modelos, tais como AlexNet, GoogLeNet e VGG16, foram selecionados com base nos resultados promissores obtidos em investigações anteriores. A partir desses testes, vamos verificar se a hipótese de diminuir o tempo de inferência e o tamanho dos modelos é válida e como essa técnica afeta o problema em questão.

5.2 Paramêtros Usados nos Modelos

Para a execução dos testes experimentais, cuja finalidade foi avaliar as quatorze arquiteturas de *CNN*s selecionadas, empregou-se a técnica de aprendizagem por transferência (*Transfer Learning*), na qual os modelos são previamente treinados em uma determinada base de dados e, posteriormente, o conjunto de pesos obtidos serve como ponto de partida para conjuntos de imagens mais específicos. As imagens da base de dados utilizada foram normalizadas e redimensionadas conforme as especificações de cada modelo.

O conjunto de dados utilizado nos experimentos foi dividido da seguinte forma: 75% para treinamento e 25% para teste. As imagens de treino foram submetidas a rotações horizontais aleatórias durante o processo de treinamento. O tamanho do lote (*batch size*) utilizado foi de 16, a taxa de aprendizado (*learning rate*) foi de 0,0003, o otimizador empregado foi o RMSprop (*Root Mean Square Propagation*) e o número de épocas especificado foi 15.

O treinamento foi conduzido na plataforma *Google Colab*, utilizando Python 3 e o *backend* do *Google Compute Engine* com 12,7 *gigabyte* (*GB*) de *random access memory* (*RAM*) e 15 *GB* de *RAM* de *GPU*. No momento da realização dos experimentos, a biblioteca *Pytorch* não permitia a execução de inferência em modelos quantizados utilizando *GPU* (KRISHNAMOORTHI *et al.*, 2024). Consequentemente, os testes de inferência apresentados na Tabela 11 , em

todas as suas variações (*float32*, *float16* e *int8*), foram executados em uma *CPU* Ryzen 4600G com 16 *GB* de *RAM*. Para a implementação do código, utilizaram-se as bibliotecas *Pytorch* 2.2.0 para a etapa de classificação de imagens e a *Scikit-learn (sklearn)* 1.4.1 para o cálculo das métricas.

Na realização deste trabalho, foi utilizada a técnica de quantização dinâmica. Após o treinamento, cada um dos 14 modelos selecionados foi convertido para um modelo quantizado. Os modelos foram quantizados usando os tipos *float16* e *int8*. No total, 42 experimentos foram realizados.

5.3 Métricas de Desempenho

Para avaliar o desempenho de classificação da base de dados das abelhas, foram utilizadas várias métricas tendo como base a matriz de confusão conforme a Tabela 8.

Tabela 8 – Exemplo de Matriz de Confusão para classificação de abelhas portadoras de pólen e abelhas não portadoras de pólen.

		Classe Prevista	
		Abelhas com pólen	**Abelhas sem pólen**
Classe Real	**Abelhas com pólen**	Verdadeiro positivo (VP)	Falso negativo (FN)
	Abelhas sem pólen	Falso positivo (FP)	Verdadeiro negativo (VN)

- Verdadeiro Positivo (VP): Número de imagens de abelhas com pólen que foram classificadas corretamente como imagens de abelhas portadoras de pólen.
- Falso Negativo (FN): Número de imagens de abelhas com pólen que foram classificadas incorretamente como imagens de abelhas não portadoras de pólen.
- Falso Positivo (FP): Número de imagens de abelhas sem pólen que foram classificadas incorretamente como imagens de abelhas portadoras de pólen.
- Verdadeiro Negativo (VN): Número de imagens de abelhas sem pólen que foram classificadas corretamente como imagens de abelhas não portadoras de pólen.

As métricas calculadas conforme a matriz de confusão foram as seguintes:

- Acurácia: Calcula a probabilidade geral de acerto. Dessa forma, a acurácia considera os acertos das duas classes, abelhas portadoras de pólen e abelhas não portadoras de pólen, sob todos os erros e acertos.

$$\text{Acurácia} = \frac{VP+VN}{VP+VN+FP+FN} \tag{5.1}$$

- Precisão: Calcula a proporção de previsões da classe de imagem positiva, que no caso estudado correspondem às abelhas não portadoras de pólen, quantas realmente eram de abelhas não portadoras de pólen.

$$\text{Precisão} = \frac{VP}{VP+FP} \tag{5.2}$$

- Revocação: Calcula a probabilidade de identificar corretamente a classe de imagem positiva, ou seja, das imagens de abelhas não portadoras de pólen, quantas foram previstas corretamente.

$$\text{Revocação} = \frac{VP}{VP+FN} \tag{5.3}$$

- *F1-score*: Faz um balanço comparando a precisão e a revocação. É calculado através da média harmônica de precisão e revocação, fornecendo um resultado que equilibra essas duas grandezas.

$$F1\text{-}score = \frac{2 \times \text{Revocação} \times \text{Precisão}}{\text{Revocação} + \text{Precisão}} \tag{5.4}$$

- Taxa de falso positivo (TFP): Calcula a proporção de classe negativa, que no caso estudado corresponde às abelhas portadoras de pólen, que são incorretamente previstas como imagem de abelhas não portadoras de pólen.

$$TFP = \frac{FP}{FP+VN} \tag{5.5}$$

5.4 Conjunto de Dados

O conjunto de dados (*dataset*) utilizado neste trabalho foi desenvolvido em 2018 por (RODRIGUEZ *et al.*, 2018). Este conjunto de dados é composto por imagens de alta resolução (180×300) de abelhas transportando e não transportando pólen, com seus respectivos rótulos, conforme ilustrado na Figura 8.

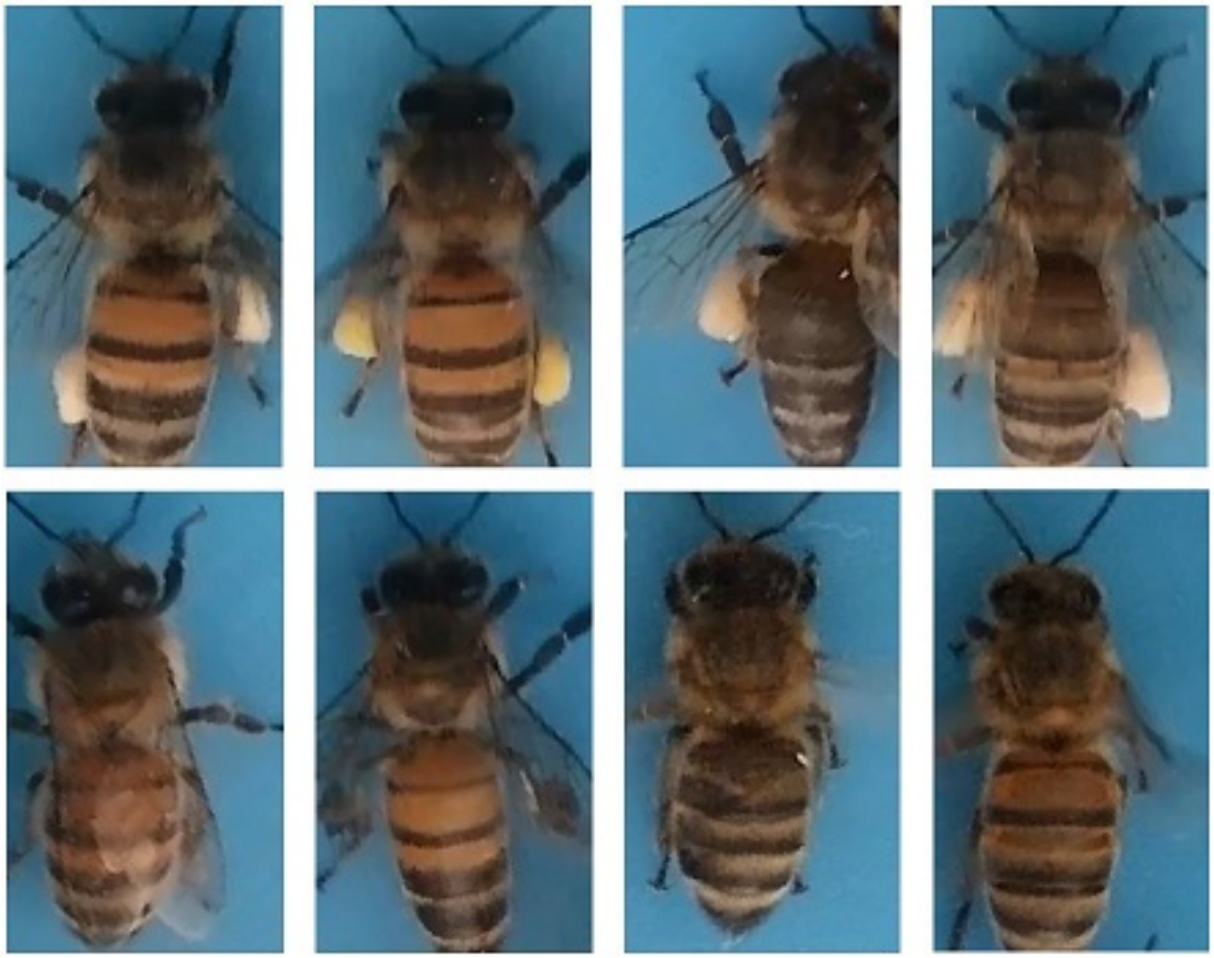


Figura 8 – Exemplos de imagens das abelhas: na primeira linha abelhas com pólen e na segunda sem pólen

Ele contém um total de 714 imagens de abelhas anotadas, das quais 369 são rotuladas como transportando pólen (*pollen bearing*) e 345 como não transportando pólen (*not pollen*). Ressalta-se que este foi o primeiro conjunto de dados público a utilizar iluminação natural, imagens de boa resolução e anotação manual da posição e orientação das abelhas[1].

[1] https://github.com/piperod/PollenDataset

As imagens foram obtidas a partir de vídeos gravados na entrada da colmeia. O vídeo era pausado assim que uma abelha com pólen entrava na rampa, então a anotação era realizada com a abelha portadora de pólen e com uma segunda abelha sem pólen no mesmo quadro, pois a rampa contém várias abelhas ao mesmo tempo, para levar em conta condições de iluminação semelhantes e garantir um conjunto de dados equilibrado. As imagens foram produzidas de modo que a orientação da abelha esteja voltada para cima e o tórax esteja centralizado, para manter a padronização. Elas também foram recortadas em um retângulo de 180×300 *pixels* e contêm uma abelha por imagem.

Seguindo a metodologia de (LE *et al.*, 2023), foram removidas algumas imagens que estavam rotuladas incorretamente. Com o objetivo de reduzir o desbalanceamento entre as classes e ampliar o efeito da rotação horizontal nas imagens, foram duplicadas 7 imagens de treinamento da classe sem pólen, resultando em um conjunto de dados com 715 imagens, sendo 365 rotuladas como “portadoras de pólen” e 350 como “não portadoras de pólen”.

6 RESULTADOS E DISCUSSÃO

Este capítulo dedica-se à apresentação dos experimentos e dos resultados obtidos nestes testes. Serão comparados o desempenho das 14 arquiteturas e uma avaliação de como a quantização afetou os tempos de inferência e o tamanho dos modelos no problema de classificação de abelhas portadoras e não portadoras de pólen.

6.1 Resultados Experimentais

Os resultados da classificação estão resumidos na Tabela 9, na qual o melhor valor para cada métrica de desempenho é indicado em negrito. Para manter uma apresentação compacta, apenas os nomes dos modelos foram incluídos, uma vez que as três variações testadas (*float32*, *16-bit floating point number* / número de ponto flutuante de 16 *bits* (*float16*) e *int8*) produziram valores idênticos de acurácia, precisão, revocação, *F1-score* e taxa de falso positivo (TFP). A título de exemplo, o modelo AlexNet alcançou consistentemente uma acurácia de 0,98 e uma precisão de 0,98 em todas as suas variações (*float32*, *float16* e *int8*). Conclui-se, portanto, que a quantização não afetou os valores dessas métricas.

Tabela 9 – Resultados de classificação,

modelo	Acurácia	Precisão	Revocação	F1-score	Taxa de falso positivo
AlexNet	0,98	0,98	0,99	0,98	0,02
EfficientNet-B0	0,98	0,96	**1**	0,98	0,04
EfficientNet-B1	0,95	0,91	**1**	0,96	0,09
GoogLeNet	0,98	0,97	**1**	0,98	0,03
MnasNet0_5	0,72	0,63	**1**	0,78	0,54
MnasNet1_3	**0,99**	**0,99**	**1**	**0,99**	**0,01**
MobileNetV2	0,98	0,97	**1**	0,98	0,03
MobileNetV3-Large	0,97	0,94	**1**	0,97	0,05
MobileNetV3-Small	0,97	0,94	**1**	0,97	0,05
RegNet-X400MF	0,95	0,91	**1**	0,96	0,09
RegNet-Y1_6GF	0,97	0,93	**1**	0,97	0,07
RegNet-Y400MF	**0,99**	**0,99**	**1**	**0,99**	**0,01**
RegNet-Y800MF	0,97	0,94	**1**	0,97	0,05
VGG16	0,98	0,97	**1**	0,98	0,03

Fonte: O autor.

A Tabela 10 contém um resumo das variações e dos menores e maiores resultados obtidos nas métricas.

Tabela 10 – Variações nas métricas com o maiores e menores resultados

Métrica	Variação	Menor resultado	Maior resultado
Acurácia	0,72 a 0,99	MnasNet0_5	RegNet-Y400MF e MnasNet1_3
Revocação	0,99 a 1	AlexNet	Todos os demais
Precisão	0,63 a 0,99	MnasNet0_5	RegNet-Y400MF e MnasNet1_3
F1-score	0,78 a 0,99	MnasNet0_5	RegNet-Y400MF e MnasNet1_3
FPR	0,01 a 0,54	RegNet-Y400MF e MnasNet1_3	MnasNet0_5

Fonte: O autor.

As Tabelas 11 e 12 apresentam os resultados dos 42 testes realizados, onde o melhor valor para cada métrica de desempenho é indicado em negrito. Na coluna de “modelo” está o modelo original em *float32* seguido pelo modelo quantizado em *int8* que é composto pelo nome do modelo com a terminação *int8* e pelo modelo quantizado em *float16* que é composto pelo nome do modelo com a terminação *float16*.

A coluna “Tempo de Teste” na Tabela 11 representa o tempo total requerido por cada modelo para processar as 176 imagens de teste, enquanto a coluna “Tempo Médio de Teste” exibe o tempo médio necessário para a inferência de uma única imagem. A coluna “Diferença de Tempo em Relação ao Original” mostra a variação percentual no tempo de inferência entre os modelos quantizados (*float16* ou *int8*) e o modelo original *float32*. Os valores de “Tempo de Teste” e “Tempo Médio de Teste” apresentados na Tabela 11 ilustram apenas um cenário possível, uma vez que os tempos variaram entre diferentes execuções. Isso pode decorrer do fato de a máquina possuir múltiplos processos em paralelo competindo pelo uso da CPU. A coluna ”Diferença de Tamanho em Relação ao Original” informa a diferença percentual no tamanho do modelo entre o modelo original e suas variações quantizadas. Para as colunas ”Diferença de Tempo em Relação ao Original” e ”Diferença de Tamanho em Relação ao Original”, valores negativos significam reduções em relação ao modelo original *float32*. Essas reduções evidenciam as melhorias de desempenho e o aumento da eficiência obtidos por meio da quantização para *float16* ou *int8*.

Da mesma forma na Tabela 12, o campo “Diferença de tamanho comparada ao original em %” compara a diferença percentual do modelo original com suas variações quantizadas. Nestes campos comparativos de “Diferença de tempo comparada ao original em %” e “Diferença

de tamanho comparada ao original em %" valores negativos se referem a reduções relativas ao modelo original em *float32* e suas quantizações em *float16* e *int8*.

Os valores de tempo de classificação das 176 imagens (tempo de teste) dos modelos variaram entre os valores de 2,3902 segundos a 41,6023 segundos. Os valores de média de tempo de teste, que representam o tempo médio para classificação de uma imagem, dos modelos variaram entre os valores de 0,0136 segundos a 0,2364 segundos.

6.1.1 Análise da Quantização

É interessante notar que todas as arquiteturas avaliadas e suas versões quantizadas tiveram desempenho semelhantes em relação às métricas de acurácia, precisão, revocação, *F1-score* e TFP como ilustrado na Tabela 9. Talvez esse resultado tenha ocorrido devido à natureza não tão complexa do problema investigado, uma classificação binária.

Em relação ao tempo de teste, a maioria dos modelos avaliados se beneficiou do processo de quantização melhorando seu desempenho. Apenas os modelos EfficientNet-B1 quantizado para *int8* e RegNet-Y1_6GF quantizado para *int8* tiveram desempenho pior que sua versão em *float32*, sendo 3,33% e 5,2% mais lentos, respectivamente. Esses resultados mostram que a quantização não garante um melhor tempo de inferência. Ainda em relação ao tempo de teste, a maioria dos modelos obteve um desempenho melhor quando quantizado para *float16*. Seis dos modelos testados (AlexNet, MnasNet0_5, MnasNet1_3, MobileNetV3-Small, RegNet-X400MF e RegNet-Y400MF) obtiveram melhores resultados quando quantizados para *int8*, inclusive o de melhor resultado, o AlexNet quando quantizado para *int8*, o qual apresentou um tempo de teste de 2,3902s, o menor tempo de inferência entre todos os modelos analisados. Interessante notar também que o pior resultado em relação ao tempo de inferência, mais precisamente 41,6023s, foi obtido pelo modelo EfficientNet-B1 quando quantizado para *int8*. Portanto, a quantização para *int8* não assegurou os melhores resultados em relação ao tempo de inferência. Os modelos que mais se beneficiaram em relação a redução do tempo de teste foram os modelos da família EfficientNet quando quantizados para *float16*. O EfficientNet-B0 obteve uma redução de 64,25% do tempo, sendo a maior redução percentual observada. Já o EfficientNet-B1 teve uma redução de 64,13% do tempo de teste.

Tabela 11 – Tabela com os resultados de tempo e comparações do modelo original e suas versões quantizadas.

modelo	Tempo de teste (s)	Média de tempo de teste	Diferença de tempo comparada ao original em %
AlexNet	4,3961	0,025	-
AlexNet int8	**2,3902**	**0,0136**	-45,63
AlexNet float16	3,1421	0,0179	-28,52
EfficientNet-B0	32,7901	0,1863	-
EfficientNet-B0 int8	27,1998	0,1545	-17,05
EfficientNet-B0 float16	11,7209	0,0666	**-64,25**
EfficientNet-B1	40,2625	0,2288	-
EfficientNet-B1 int8	41,6023	0,2364	3,33
EfficientNet-B1 float16	14,4434	0,0821	-64,13
GoogLeNet	9,7841	0,0556	-
GoogLeNet int8	9,623	0,0547	-1,65
GoogLeNet float16	9,6046	0,0546	-1,83
MnasNet0_5	3,9081	0,0222	-
MnasNet0_5 int8	3,6046	0,0205	-7,77
MnasNet0_5 float16	3,756	0,0213	-3,89
MnasNet1_3	7,1776	0,0408	-
MnasNet1_3 int8	6,6175	0,0376	-7,8
MnasNet1_3 float16	7,0074	0,0398	-2,37
MobileNetV2	16,4975	0,0937	-
MobileNetV2 int8	13,8495	0,0787	-16,05
MobileNetV2 float16	9,0975	0,0517	-44,86
MobileNetV3-Large	6,6647	0,0379	-
MobileNetV3-Large int8	6,461	0,0367	-3,06
MobileNetV3-Large float16	6,1732	0,0351	-7,38
MobileNetV3-Small	3,6438	0,0207	-
MobileNetV3-Small int8	3,2061	0,0182	-12,01
MobileNetV3-Small float16	3,5005	0,0199	-3,93
RegNet-X400MF	5,2535	0,0298	-
RegNet-X400MF int8	4,9288	0,028	-6,18
RegNet-X400MF float16	5,1895	0,0295	-1,22
RegNet-Y1_6GF	14,5459	0,0826	-
RegNet-Y1_6GF int8	15,3027	0,0869	5,2
RegNet-Y1_6GF float16	13,0503	0,0741	-10,28
RegNet-Y400MF	6,6522	0,0378	-
RegNet-Y400MF int8	5,6538	0,0321	-15,01
RegNet-Y400MF float16	5,972	0,0339	-10,23
RegNet-Y800MF	7,7305	0,0439	-
RegNet-Y800MF int8	7,7032	0,0438	-0,35
RegNet-Y800MF float16	6,7945	0,0386	-12,11
VGG16	31,9778	0,1817	-
VGG16 int8	27,3269	0,1553	-14,54
VGG16 float16	24,7501	0,1406	-22,6

Fonte: O autor.

Tabela 12 – Tabela com os resultados de tamanho e comparações do modelo original e suas versões quantizadas.

modelo	Tamanho do modelo (KB)	Diferença de tamanho comparada ao original em %
AlexNet	228037,87	-
AlexNet int8	64449,816	**-71,7372**
AlexNet float16	228039,832	0,0009
EfficientNet-B0	16335,098	-
EfficientNet-B0 int8	16332,122	-0,0182
EfficientNet-B0 float16	16335,898	0,0049
EfficientNet-B1	26491,898	-
EfficientNet-B1 int8	26488,922	-0,0112
EfficientNet-B1 float16	26492,634	0,0028
GoogLeNet	22581,394	-
GoogLeNet int8	22579,174	-0,0098
GoogLeNet float16	22582,182	0,0035
MnasNet0_5	3942,436	-
MnasNet0_5 int8	**3939,45**	-0,0757
MnasNet0_5 float16	3943,162	0,0184
MnasNet1_3	20309,476	-
MnasNet1_3 int8	20306,49	-0,0147
MnasNet1_3 float16	20310,202	0,0036
MobileNetV2	9145,312	-
MobileNetV2 int8	9142,394	-0,03
MobileNetV2 float16	9146,106	0,01
MobileNetV3-Large	17020,654	-
MobileNetV3-Large int8	13332,09	-21,67
MobileNetV3-Large float16	17022,202	0,01
MobileNetV3-Small	6210,85	-
MobileNetV3-Small int8	4439,982	-28,51
MobileNetV3-Small float16	6212,398	0,02
RegNet-X400MF	20694,986	-
RegNet-X400MF int8	20694,632	-0,0017
RegNet-X400MF float16	20695,72	0,0035
RegNet-Y1_6GF	41708,004	-
RegNet-Y1_6GF int8	41706,178	-0,0044
RegNet-Y1_6GF float16	41708,738	0,0018
RegNet-Y400MF	15867,638	-
RegNet-Y400MF int8	15867,156	-0,003
RegNet-Y400MF float16	15868,372	0,0046
RegNet-Y800MF	22846,754	-
RegNet-Y800MF int8	22845,248	-0,0066
RegNet-Y800MF float16	22847,488	0,0032
VGG16	537069,974	-
VGG16 int8	178447,028	-66,774
VGG16 float16	537072,18	0,0004

Fonte: O autor.

Na Figura 9 podemos visualizar as 5 maiores reduções no tempo de inferência do modelos quantizados em relação aos modelos em suas versões em *float32*.

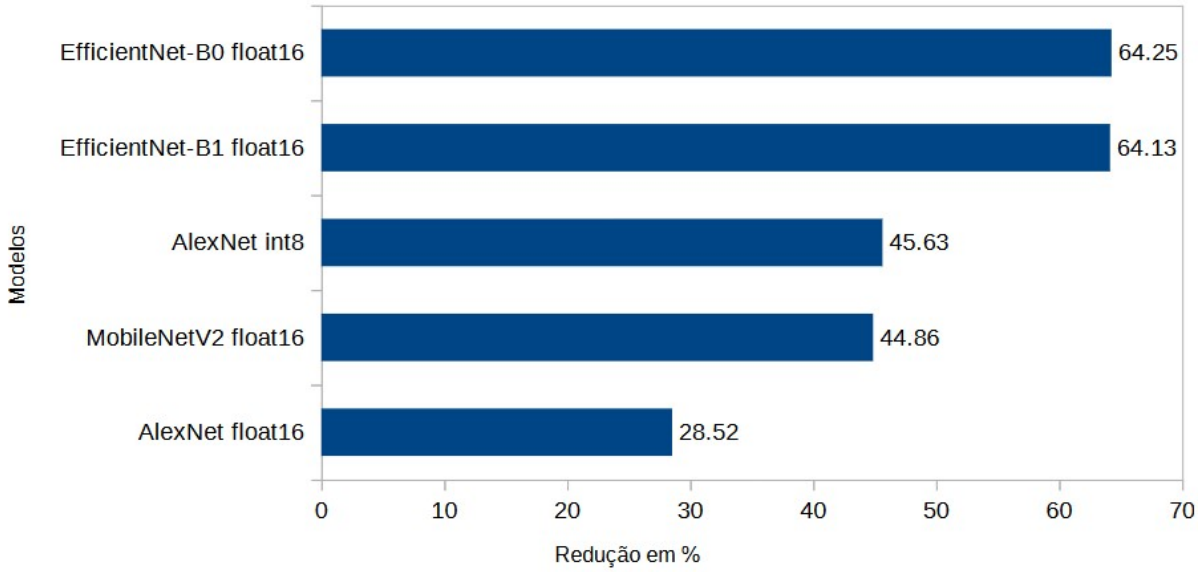


Figura 9 – Top 5 maiores reduções no tempo.
Fonte: O autor

Na Figura 10 podemos visualizar os 5 menores tempos de inferência. Podemos observar quem estes 5 menores tempos foram obtidos por modelos quantizados.

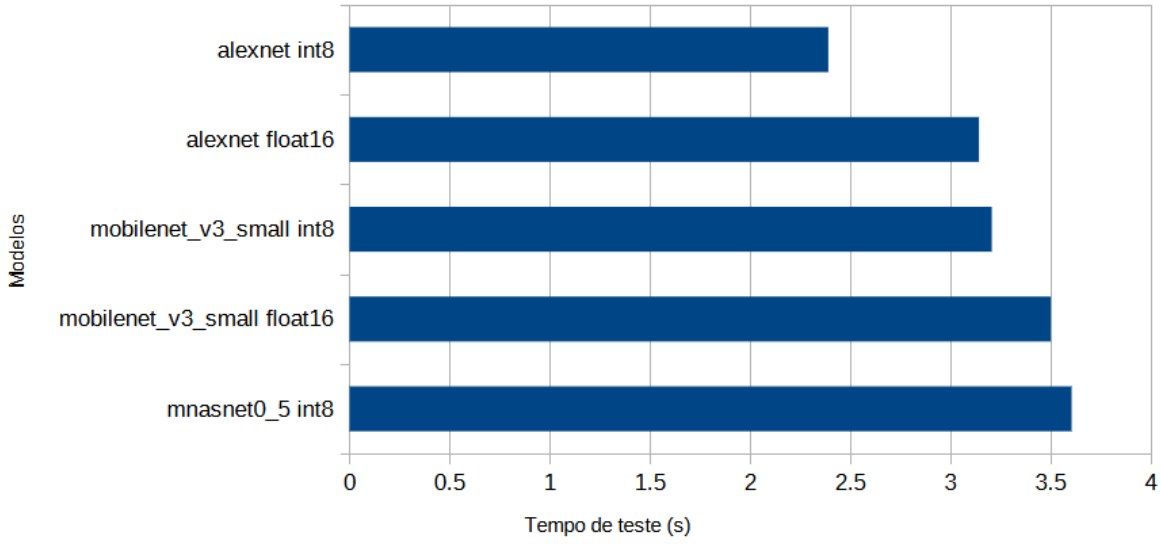


Figura 10 – Menores tempos de inferência.
Fonte: O autor

Em relação ao tamanho dos modelos, o de menor tamanho foi o MnasNet0_5, com 3942,436 *kilobyte* (*KB*). Contudo, este modelo também apresentou o pior desempenho no que tange à acurácia, à precisão, à *F1-score* e à TFP. Todos os modelos sofreram reduções de tamanho

ao serem quantizados para *int8*, em comparação com suas contrapartes em *float32*. Para alguns modelos, incluindo EfficientNet-B0, EfficientNet-B1, GoogLeNet, MnasNet0_5, MnasNet1_3, MobileNetV2, RegNet-X400MF, RegNet-Y1_6GF, RegNet-Y400MF e RegNet-Y800MF, as reduções foram mínimas. Em contrapartida, outros modelos alcançaram reduções de tamanho significativas por meio da quantização para *int8*. Notavelmente, o modelo VGG16 obteve uma redução de tamanho de 66,774%, enquanto o modelo AlexNet alcançou a maior redução, com um decréscimo de 71,7372% em comparação com seu tamanho original. Por outro lado, a quantização para *float16* resultou em um ligeiro aumento de tamanho em todos os modelos avaliados. Estes resultados demonstram que, embora a quantização para *int8* possa ser altamente eficaz na redução do tamanho dos modelos, a quantização para *float16* pode levar a aumentos de tamanho negligenciáveis.

Na Figura 11 podemos visualizar as 5 maiores reduções no tamanho dos modelos quantizados em relação aos modelos em suas versões em *float32*. Na Figura 12 podemos visualizar os 5 menores tamanhos de modelos obtidos.

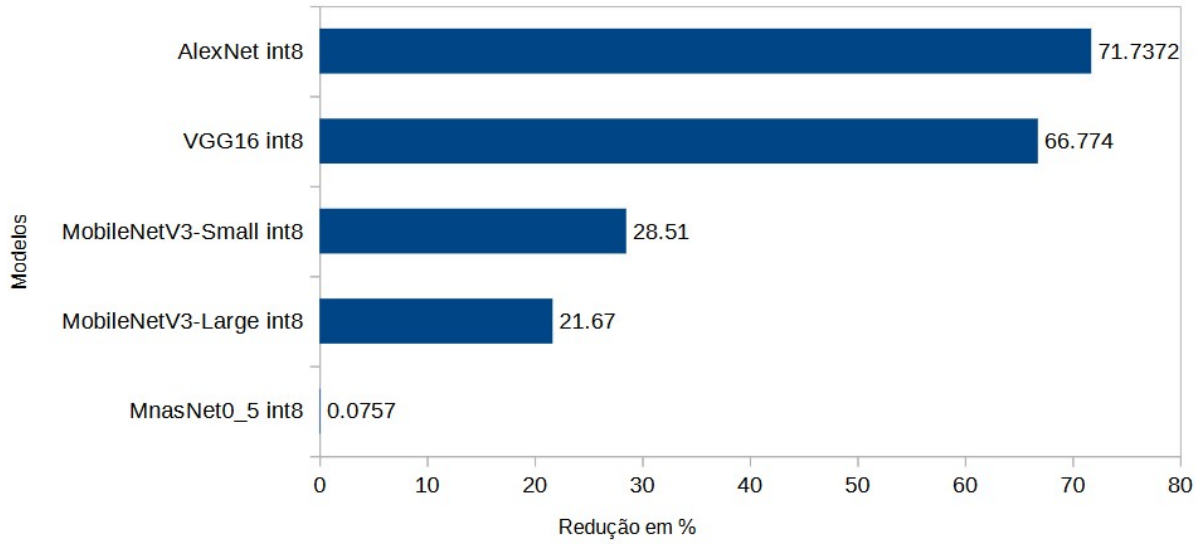


Figura 11 – Top 5 maiores reduções de tamanho.
Fonte: O autor

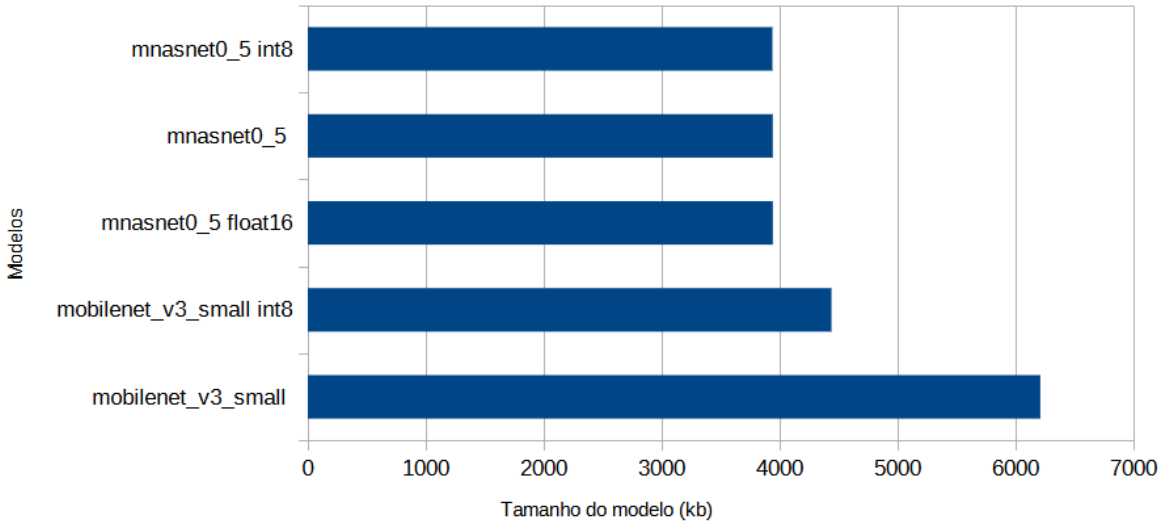


Figura 12 – Menores tamanhos.
Fonte: O autor

De forma geral, o modelo que obteve os melhores resultados no processo de quantização foi o AlexNet. Além de manter o desempenho em relação à acurácia, o AlexNet quando quantizado para *int8* teve o menor tempo de inferência dentre todos os modelos. Ele também conseguiu alcançar expressivas reduções no tempo de inferência (-45,63%) e no tamanho do modelo (-71,7372%).

As reduções no tempo de inferência e no tamanho dos modelos mostram que a quantização pode ser o caminho para a viabilidade de um projeto que utilize equipamentos com restrições no tamanho de memória e no poder de processamento. Além disso, os resultados mostram que,no problema abordado, existem casos com redução no tempo de inferência e no tamanho do modelo sem afetar os resultados de métricas como a acurácia.

6.2 Segundo Experimento

Dada a limitação de *hardware* do primeiro experimento, um segundo foi conduzido com configurações modificadas e mais rodadas de testes. Neste experimento, foi aplicada uma abordagem k-fold com 5 *folds*. O tamanho do lote foi reduzido para 8, a taxa de aprendizado foi definida como 0,0001 e o otimizador permaneceu RMSprop. O número de épocas de treinamento foi definido como 10. O treinamento foi realizado em um Intel i9 com 32 *GB* de *RAM* e uma RTX 4070 com 12 *GB* de *video random access memory* (*VRAM*). Para cada modelo, cinco execuções de treinamento foram realizadas, e a de melhor desempenho foi selecionada para avaliação no conjunto de teste.

Na Tabela 13, apenas os nomes dos modelos são apresentados, uma vez que as versões *float32*, *float16* e *int8* obtiveram novamente resultados idênticos. Curiosamente, nesta segunda rodada de treinamento, algumas famílias de modelos alcançaram valores idênticos em todas as métricas. Em particular, os modelos EfficientNet-B0 e EfficientNet-B1, bem como RegNet-Y400MF e RegNet-Y800MF, alcançaram os mesmos níveis de acurácia, precisão, revocação e *F1-score*.

Tabela 13 – Resultados da classificação do segundo experimento.

Modelo	Acurácia	Precisão	Revocação	F1-score
AlexNet	0,96	0,94	0,98	0,96
EfficientNet-B0	**0,98**	**0,96**	**1,00**	**0,98**
EfficientNet-B1	**0,98**	**0,96**	**1,00**	**0,98**
GoogLeNet	0,96	0,93	**1,00**	0,96
MnasNet0_5	0,64	0,57	0,98	0,73
MnasNet1_3	**0,98**	**0,96**	**1,00**	**0,98**
MobileNetV2	0,94	0,90	**1,00**	0,94
MobileNetV3-Large	0,97	0,94	**1,00**	0,97
MobileNetV3-Small	0,94	0,89	**1,00**	0,94
RegNet-X400MF	0,96	0,93	0,98	0,96
RegNet-Y1_6GF	0,96	0,93	**1,00**	0,96
RegNet-Y400MF	0,96	0,92	**1,00**	0,96
RegNet-Y800MF	0,96	0,92	**1,00**	0,96
VGG16	0,94	0,90	**1,00**	0,94

Também medimos o tamanho dos modelos nesta segunda rodada de testes, e os resultados são apresentados na Tabela 14. Os resultados foram próximos aos relatados na Tabela 12, indicando tendências semelhantes na redução do tamanho do modelo após a quantização.

Tabela 14 – Resultados de tamanho do segundo experimento.

Modelo	Tamanho do modelo (KB)
AlexNet	228038,287
AlexNet int8	64450,289
AlexNet float16	228040,249
EfficientNet-B0	16335,531
EfficientNet-B0 int8	16333,355
EfficientNet-B0 float16	16336,331
EfficientNet-B1	26492,331
EfficientNet-B1 int8	26490,387
EfficientNet-B1 float16	26493,067
GoogLeNet	22581,879
GoogLeNet int8	22580,299
GoogLeNet float16	22582,603
MnasNet0_5	3942,859
MnasNet0_5 int8	**3940,581**
MnasNet0_5 float16	3943,585
MnasNet1_3	20309,899
MnasNet1_3 int8	20307,621
MnasNet1_3 float16	20310,625
MobileNetV2	9145,803
MobileNetV2 int8	9143,465
MobileNetV2 float16	9146,533
MobileNetV3-Large	17021,093
MobileNetV3-Large int8	13333,173
MobileNetV3-Large float16	17022,641
MobileNetV3-Small	6211,289
MobileNetV3-Small int8	4440,929
MobileNetV3-Small float16	6212,837
RegNet-X400MF	20695,417
RegNet-X400MF int8	20695,935
RegNet-X400MF float16	20696,215
RegNet-Y1_6GF	41708,435
RegNet-Y1_6GF int8	41707,877
RegNet-Y1_6GF float16	41709,169
RegNet-Y400MF	15868,069
RegNet-Y400MF int8	15868,371
RegNet-Y400MF float16	15868,803
RegNet-Y800MF	22847,185
RegNet-Y800MF int8	22846,375
RegNet-Y800MF float16	22847,919
VGG16	537070,451
VGG16 int8	178447,529
VGG16 float16	537072,593

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste estudo deixamos como contribuição a analise de quatorze modelos de redes neurais convolucionais profundas para a classificação de abelhas portadoras de pólen versus aquelas sem pólen em imagens obtidas na entrada da colmeia. Além disso, investigamos os efeitos do processo de quantização no desempenho desses modelos. A quantização pode reduzir o tempo de inferência e o tamanho do modelo, utilizando formatos de menor precisão para cálculos e armazenamento de dados. Considerando o uso frequente de sistemas embarcados para aquisição de imagens, que muitas vezes são limitados em memória e recursos computacionais, os modelos quantizados oferecem vantagens notáveis. Nossos resultados mostram que, para o conjunto de dados avaliado, é possível melhorar o tempo de inferência e/ou o tamanho do modelo sem comprometer as principais métricas de desempenho, como acurácia, precisão, revocação, *F1-score* e taxa de falsos positivos (TFP).

Considerando todos os resultados obtidos, não há um modelo que seja superior em todos os aspectos em relação aos outros. Dessa forma, deve-se levar em consideração as prioridades em um projeto de implantação, sendo fundamental o teste das opções disponíveis para uma escolha mais adequada ao problema abordado. Caso a prioridade seja alguma das métricas de acurácia, precisão, revocação, *F1-score* ou TFP a escolha seria MnasNet1_3 quantizado em *int8* ou RegNet-Y400MF quantizado em *int8*. Entre eles, RegNet-Y400MF tem uma ligeira vantagem devido ao melhor tempo de inferência e ao menor tamanho do modelo. Caso o fator crítico seja o tempo de inferência, a melhor escolha seria o AlexNet quantizado em *int8*. Ele alcançou o menor tempo de inferência, mantendo um bom desempenho geral, com uma acurácia de 0,98. No caso de se priorizar o tamanho do modelo, o melhor caso foi do modelo MnasNet0_5 quantizado em *int8*. Entretanto, sua baixa acurácia de 0,72 não favorece sua escolha. Uma alternativa melhor pode ser o MobileNetV3-Small quantizado para *int8*, que apresenta tamanho reduzido (4439,982 *KB*, em comparação com 3942,436 *KB* para MnasNet0_5) e alcança acurácia maior de 0,97. Esse equilíbrio entre tamanho e acurácia o torna uma opção mais prática para cenários em que ambos são importantes.

É possível concluir também que os modelos de classificação podem se beneficiar bastante dos processos de quantização, pois foram atingidas reduções de até 64,25% no tempo de inferência e de 71,73% no tamanho sem interferir em outras métricas de desempenho como a acurácia. O maior beneficiado do processo de quantização foi o modelo AlexNet quando quantizado para *int8*, pois alcançou grandes reduções no tempo de inferência (-45.63%) e no

tamanho do modelo (-71.7372%). Assim podemos atestar que o objetivo de avaliar o impacto da quantização foi alcançado.

Como trabalho futuro, planejamos avaliar conjuntos de dados adicionais. Prevemos que a realização de testes mais abrangentes podem fornecer uma visão mais profunda sobre as limitações dos modelos, o que, em última análise, levará a um melhor desempenho de classificação. Pretendemos analisar também como a quantização pode influenciar na etapa de treinamento. Além disso, planejamos avaliar diferentes técnicas de quantização e testar seu impacto em dispositivos de borda.

REFERÊNCIAS

BABIC, Z.; PILIPOVIC, R.; RISOJEVIC, V.; MIRJANIC, G. Pollen bearing honey bee detection in hive entrance video recorded by remote embedded system for pollination monitoring. **ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, Copernicus GmbH, v. 3, p. 51–57, 2016.

BERKAYA, S. K.; GUNAL, E. S.; GUNAL, S. Deep learning-based classification models for beehive monitoring. **Ecological Informatics**, Elsevier, v. 64, p. 101353, 2021.

BILIK, S.; ZEMCIK, T.; KRATOCHVILA, L.; RICANEK, D.; RICHTER, M.; ZAMBANINI, S.; HORAK, K. Machine learning and computer vision techniques in continuous beehive monitoring applications: A survey. **Computers and Electronics in Agriculture**, Elsevier, v. 217, p. 108560, 2024.

CHILAMKURTHY, S. **Transfer Learning for Computer Vision Tutorial**. 2025. Disponível em: https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html.

FOOD; NATIONS, A. O. of the U. **Pollinators vital to our food supply under threat**. 2016. Disponível em: https://www.fao.org/newsroom/detail/Pollinators-vital-to-our-food-supply-under-threat/en.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

HADJUR, H.; AMMAR, D.; LEFÈVRE, L. Toward an intelligent and efficient beehive: A survey of precision beekeeping systems and services. **Computers and Electronics in Agriculture**, Elsevier, v. 192, p. 106604, 2022.

HOLDSWORTH, M. S. J. **Deep Learning**. 2024. Disponível em: https://www.ibm.com/br-pt/topics/deep-learning.

HOWARD, A.; SANDLER, M.; CHU, G.; CHEN, L.-C.; CHEN, B.; TAN, M.; WANG, W.; ZHU, Y.; PANG, R.; VASUDEVAN, V. *et al.* Searching for mobilenetv3. In: **Proceedings of the IEEE/CVF international conference on computer vision**. [S.l.: s.n.], 2019. p. 1314–1324.

IBM. **What is Machine Learning?** 2017. Disponível em: https://www.ibm.com/think/topics/machine-learning.

IBM. **Convolutional Neural Networks**. 2024. Disponível em: https://www.ibm.com/br-pt/topics/convolutional-neural-networks.

IMAGENET. **About ImageNet**. 2024. Disponível em: https://image-net.org/about.php.

KRISHNAMOORTHI, R.; REED, J.; NI, M.; GOTTBRATH, C.; WEIDMAN, S. **Introduction to Quantization on PyTorch**. 2024. Disponível em: https://pytorch.org/blog/introduction-to-quantization-on-pytorch/.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, 2012.

LE, T.-N.; THI-THU-HONG, P.; NGUYEN, H.-D.; THI-LAN, L. *et al.* A novel convolutional neural network architecture for pollen-bearing honeybee recognition. **International Journal of Advanced Computer Science and Applications**, Science and Information (SAI) Organization Limited, v. 14, n. 8, 2023.

LEARNOPENCV. **Understanding Convolutional Neural Networks (CNN)**. 2024. Disponível em: https://learnopencv.com/understanding-convolutional-neural-networks-cnn/.

MITCHELL, T. M. **Machine Learning**. 1. ed. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077.

MONTEIRO, F. C.; PINTO, C. M.; RUFINO, J. Towards precise recognition of pollen bearing bees by convolutional neural networks. In: SPRINGER. **Iberoamerican Congress on Pattern Recognition**. [S.l.], 2021. p. 217–226.

NGO, T. N.; RUSTIA, D. J. A.; YANG, E.-C.; LIN, T.-T. Automated monitoring and analyses of honey bee pollen foraging behavior using a deep learning-based imaging system. **Computers and Electronics in Agriculture**, Elsevier, v. 187, p. 106239, 2021.

NGUYEN, D.; LE, T.; PHUNG, T.; NGUYEN, D.; NGUYEN, H.; PHAM, H.; PHAN, T.; VU, H.; LE, T. Improving pollen-bearing honey bee detection from videos captured at hive entrance by combining deep learning and handling imbalance techniques. **Ecol. Informatics**, v. 82, p. 102744, 2024. Disponível em: https://doi.org/10.1016/j.ecoinf.2024.102744.

PYTORCH. **Quantization**. 2024. Disponível em: https://pytorch.org/docs/stable/quantization.html.

RADOSAVOVIC, I.; KOSARAJU, R. P.; GIRSHICK, R.; HE, K.; DOLLÁR, P. Designing network design spaces. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [S.l.: s.n.], 2020. p. 10428–10436.

RODRIGUEZ, I. F.; MEGRET, R.; ACUNA, E.; AGOSTO-RIVERA, J. L.; GIRAY, T. Recognition of pollen-bearing bees from video using convolutional neural network. In: IEEE. **2018 IEEE winter conference on applications of computer vision (WACV)**. [S.l.], 2018. p. 314–322.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 4510–4520.

SIMONYAN, K. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SLEDEVIČ, T. The application of convolutional neural network for pollen bearing bee classification. In: IEEE. **2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)**. [S.l.], 2018. p. 1–4.

STOJNIĆ, V.; RISOJEVIĆ, V.; PILIPOVIĆ, R. Detection of pollen bearing honey bees in hive entrance images. In: IEEE. **2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)**. [S.l.], 2018. p. 1–4.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9.

TAN, M.; CHEN, B.; PANG, R.; VASUDEVAN, V.; SANDLER, M.; HOWARD, A.; LE, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [S.l.: s.n.], 2019. p. 2820–2828.

TAN, M.; LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: PMLR. **International conference on machine learning**. [S.l.], 2019. p. 6105–6114.

WU, H.; JUDD, P.; ZHANG, X.; ISAEV, M.; MICIKEVICIUS, P. Integer quantization for deep learning inference: Principles and empirical evaluation. **arXiv preprint arXiv:2004.09602**, 2020.

YANG, C.; COLLINS, J. Deep learning for pollen sac detection and measurement on honeybee monitoring video. In: IEEE. **2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)**. [S.l.], 2019. p. 1–6.

ZACEPINS, A.; BRUSBARDIS, V.; MEITALOVS, J.; STALIDZANS, E. Challenges in the development of precision beekeeping. **Biosystems Engineering**, Elsevier, v. 130, p. 60–71, 2015.