Universidade Federal do ABC
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas
Programa de Graduação em Engenharia da Informação

Estudo de algoritmos de processamento e classificação de biometrias por impressão digital

Artur Lazarini Silva

Santo André - SP

2025

Artur Lazarini Silva

Estudo de algoritmos de processamento e classificação de biometrias por impressão digital

Trabalho de Graduação apresentado ao curso de Engenharia de Informação da Universidade Federal do ABC, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia de Informação.

Universidade Federal do ABC – UFABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

Programa de Graduação em Engenharia da Informação

Orientador: André Kazuo Takahata

Santo André - SP

2025

Resumo

A busca de registros de impressões digitais em bases de dados extensas exige elevado custo computacional, especialmente em sistemas de identificação biométrica que demandam respostas rápidas. A utilização de algoritmos de classificação contribui para a organização estrutural dessas bases e para a redução significativa do tempo de busca. Este estudo implementa técnicas de reconhecimento de padrões e classificação de imagens baseadas em redes neurais convolucionais, com o objetivo de avaliar o desempenho de diferentes arquiteturas no processo de classificação de biometrias digitais. Foram analisadas as redes LeNet, AlexNet, CaffeNet e FCTP-Net, treinadas em um conjunto contendo aproximadamente 7.200 amostras biométricas. Os modelos resultantes apresentaram acurácias de validação de 81,8%, 88,6%, 88,2% e 87,3%, respectivamente. A aplicação dos modelos treinados permitiu a utilização dos classificadores em qualquer imagem de impressão digital, proporcionando maior eficiência no processo de busca. Como resultado, obteve-se uma redução média de 65% no tempo de consulta em uma base de dados composta por cerca de um milhão de registros, empregando o modelo com aproximadamente 89% de acurácia. Os resultados demonstram o potencial das redes neurais convolucionais para otimizar processos de identificação biométrica em grandes sistemas de armazenamento.

Palavras-chaves: Redes Neurais Convolucionais; Reconhecimento de imagem; Classificação de imagens; Biometria; classificação de impressões digitais.

Abstract

The search of fingerprint records in large databases is computationally demanding, particularly in biometric identification systems that require fast response times. The use of classification algorithms contributes to the structural organization of these databases and significantly reduces search time. This study implements pattern-recognition and image-classification techniques based on convolutional neural networks to evaluate the performance of different architectures for digital biometric classification. The LeNet, AlexNet, CaffeNet, and FCTP-Net networks were trained on a dataset containing approximately 7,200 biometric samples. The resulting models achieved validation accuracies of 81.8%, 88.6%, 88.2%, and 87.3%, respectively. The use of the trained models enabled the application of classifiers to any fingerprint image, thereby improving the efficiency of the search process. As a result, an average reduction of 65% in query time was achieved in a database with about one million records by employing the model with approximately 89% classification accuracy. The findings highlight the potential of convolutional neural networks to optimize biometric identification processes in large-scale storage systems.

Keywords: Convolutional Neural Networks; Image Recognition; Image Classification; Biometrics; Fingerprint classification.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UFABC	Universidade Federal do ABC
CECS	Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas
A	Impressão digital do tipo *Arch* (Arco)
TA	Impressão digital do tipo *Tented Arch* (Arco agudo)
LL	Impressão digital do tipo *Left Loop* (Laço Esquerdo)
RL	Impressão digital do tipo *Right Loop* (Laço Direito)
W	Impressão digital do tipo *Whorl* (Redemoinho)
CASIA	*Institute of Automation, Chinese Academy of Sciences*
NIST	*National Institute of Standards and Technology*
NN	*Neural Network*
DNN	*Deep Neural Network*
CNN	*Convolutional Neural Network*
ReLu	*Rectified Linear Unit*
Caffe	*Convolutional Architecture for Fast Feature Embedding*
LRN	*Local Response Normalization*
FC	*Fully Connected*
DL	*Deep Learning*
CCD	*Charge Coupled Devices*
CMOS	*Complementary Metal–Oxide–Semiconductor*
GPU	*Graphics Processing Unit*
TPU	*Tensor Processing Unit*
MCU	Microcontrolador
NPU	*Neural Processing Unit*
TF	*TensorFlow*
TFLite	*TensorFlow Lite*

Sumário

1 Introdução

A palavra biometria é derivada do grego *bios*, que significa vida, e *metron*, que significa medição. Logo esta palavra diz respeito às medidas de partes do corpo dos seres vivos, que podem ser utilizadas para diferenciação de indivíduos. A identificação por meio destas métricas combina aspectos anatômicos e comportamentais dos indivíduos, portanto não devem ser classificados exclusivamente por uma destas formas. As biometrias são naturalmente anatômicas, entretanto, o modo como os usuários as apresentam, como por exemplo o modo com que um usuário pressiona o dedo em um sensor de captura ou o modo do usuário apresentar a face em uma câmera, dependem do comportamento do usuário. Portanto as entradas em sistemas de reconhecimento dependem da combinação de características físicas e comportamentais. A identificação biométrica, por meio de suas características únicas, fornece métodos para verificar se as informações são verdadeiras, ou semelhantes a um conjunto de características armazenadas de um individuo. As métricas mais utilizadas em sistemas de reconhecimento são as faciais, íris e as impressões digitais [MALTONI et al., 2009].

As tecnologias de reconhecimento por biometrias de face para diferenciação de indivíduos, funcionam a partir de uma imagem ou um video de alta resolução. A extração de pontos de interesse que representam características da face, como as intersecções de linhas de expressões, o contorno dos olhos ou o contorno da boca e queixo. O sistema é capaz de reconhecer as informações extraídas da face pela localização de distribuição dos pontos de interesse da expressão facial. As métricas mais comuns que podem ser citadas são, a distancia entre os olhos, tamanho do nariz, profundidade das cavidades do olhos, formato de linhas de expressões das bochechas e o formato do queixo. A amostra da imagem capturada da face, segue o procedimento de detecção de pontos de interesse, recorte da região da face, extração dos pontos de interesse e finalmente a identificação da face a partir das características obtidas. O método de extração e reconhecimento de faces varia de acordo com o algoritmo e necessidade do sistema. É ilustrado na Figura 1 um exemplo de extração de pontos de interesse e características de uma face [ZHOU; LIANG; TAN, 2023].

A tecnologia, apesar de avançada e em grande expansão no mercado, possui particularidades que dificultam a acurácia da validação do usuário. Usuários que eventualmente podem utilizar acessórios, como por exemplo óculos, chapéus ou maquiagens, podem alterar a fisionomia do rosto a ponto de invalidar ou dificultar a identificação do usuário. Aspectos comportamentais, como mudanças no estilo de vida podem desencadear uma alteração na massa corpórea, podendo também alterar a fisionomia do rosto. Usuários com irmãos gêmeos, podem ser confundidos pelo algoritmo, por possuírem características

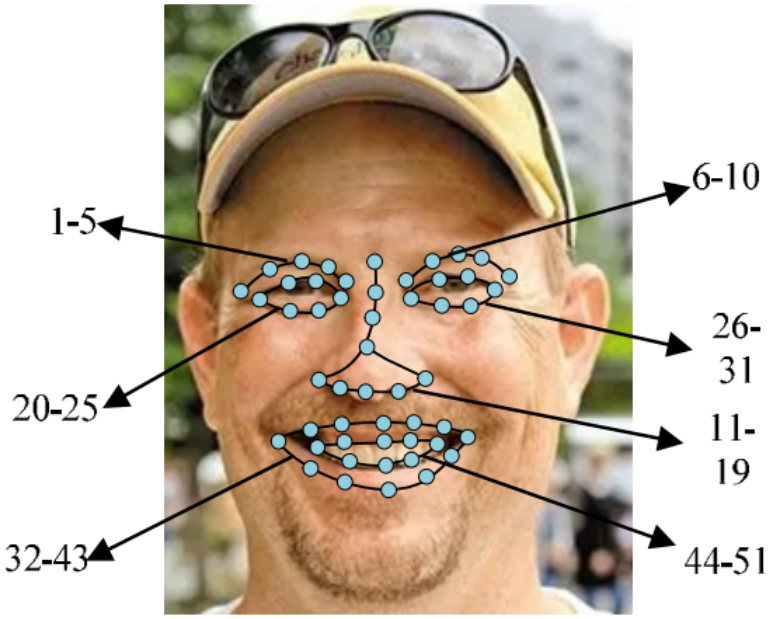


Figura 1 – Captura de pontos de verificação facial. Fonte: [ZHOU; LIANG; TAN, 2023]

faciais iguais ou muito semelhantes entre si [MALTONI et al., 2009].

A tecnologia de reconhecimento por íris funciona a partir de uma imagem ou vídeo de alta resolução, de forma semelhante aos algoritmos de reconhecimento facial. A Figura 2 ilustra um algoritmo que captura os contornos internos e externos da íris para identificação. Diferentemente do reconhecimento facial, o reconhecimento pela íris requer um posicionamento e orientação espacial precisos dos olhos do usuário em relação à câmera de captura, o que dificulta a sua utilização em grande escala [DAUGMAN, 2007].

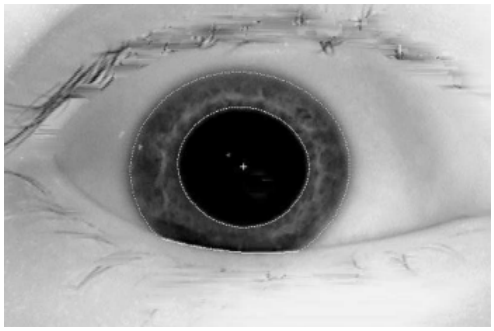


Figura 2 – Captura de pontos de verificação por iris. Fonte: [DAUGMAN, 2007]

A tecnologia de reconhecimento por digitais, funciona a partir de uma imagem de baixa ou média resolução. A Figura 3 exemplifica o procedimento que captura os picos e vales formados pela epiderme do dedos, que compõem os padrões das biometrias digitais. Por utilizar câmeras de baixa resolução e por ter uma maior facilidade de uso, a identificação por impressão digital ainda se mantém como um método confiável, de fácil implementação e baixo custo [JAIN; ROSS; PRABHAKAR, 2004].

A íris e as digitais dos dedos são padrões de identificação únicos, mesmo entre

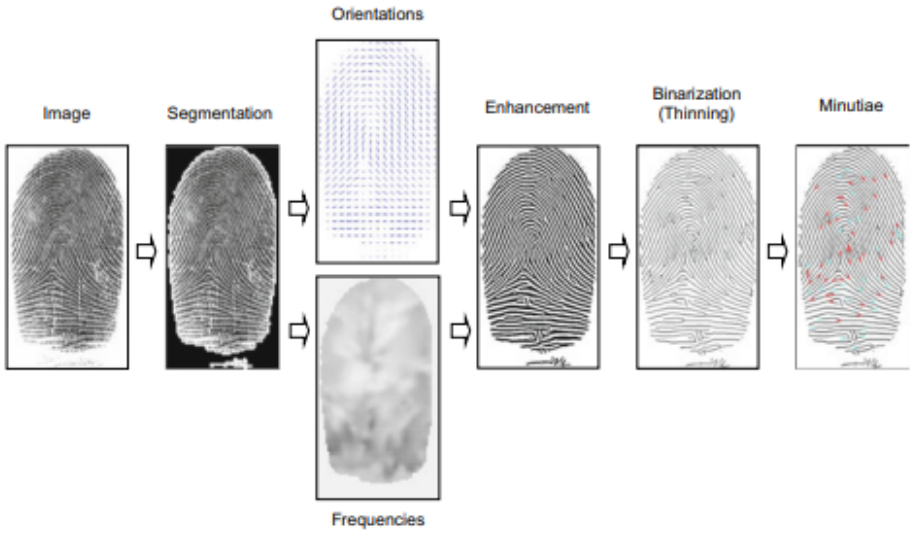


Figura 3 – Fluxo de extração de características de uma impressão digital. Fonte: [MALTONI et al., 2009]

gêmeos idênticos. A formação dessas duas biometrias começa no estágio fetal dos indivíduos. Durante a fase de diferenciação dos tecidos, pequenas variações em microambientes do útero, como a posição e o fluxo do liquido amniótico ao redor do feto, desencadeiam alterações no processo de crescimento da epiderme e, consequentemente, criam variações de mão em mão, dedo por dedo, feto por feto. Sendo assim, é virtualmente impossível que duas biometrias digitais sejam exatamente iguais. Entretanto, por terem fatores genéticos, algumas características ou padrões serão semelhantes entre familiares [MALTONI et al., 2009].

Sistemas que utilizam *Deep learning* (DL), ou aprendizado profundo, são utilizados, de forma simplificada, para a análise e diferenciação de imagens em duas dimensões. As diversas camadas ocultas aprendem características dos dados brutos de entrada e como resultado de uma função de perda, determinam as predições do sistema. Independentemente dos padrões de características, formatos ou pontos, o DL define um algoritmo que analisa as imagens bidimensionais por meio de sua distribuição de cores. Adicionalmente, uma grande contribuição da utilização do DL é que esses sistemas de aprendizado necessitam de uma grande quantidade de tempo durante o treinamento dos modelos, dependendo da capacidade do computador, da quantidade de dados e dos parâmetros de treino. Entretanto, a implementação do modelo treinado permite uma rápida execução do processo de detecção, possibilitando que os sistemas de DL sejam o estado da arte para métodos de classificação em tempo real [RIM; KIM; HONG, 2021].

Há inúmeras vantagens em migrar sistemas de processamento em tempo real baseados na nuvem para dispositivos com redes neurais embarcadas, principalmente considerando a privacidade dos dados e a segurança da informação. Diferente dos sistemas em nuvem, a inferência realizada localmente reduz os riscos de segurança ao processar

dados sensíveis diretamente no dispositivo, reduzindo também a latência, ou o tempo de resposta, durante a operação. Este tipo de processamento local é chamado de *edge computing*, ou computação na borda.

Entretanto, dispositivos com aceleradores de redes neurais, como *Graphics Processing Unit* (GPU) e *Tensor Processing Unit* (TPU), consomem espaço e energia em ambientes com recursos restritos. Microcontroladores (MCUs) são dispositivos compactos, de baixo custo e baixo consumo energético, geralmente dependentes de um único núcleo de processamento e memória compartilhada entre seus periféricos internos. MCUs são utilizados em aplicações com recursos limitados e comumente são incapazes de realizar inferências de redes neurais. A falta de *hardware* interno especifico para aceleração de processamento resulta em uma grande latência e um consumo elevado de energia. As limitações de memória, processamento e armazenamento aumentam o desafio de gerir as grandes matrizes de pesos necessárias para as inferências de redes neurais. Como resposta às limitações dos microcontroladores tradicionais, a industria passou a implementar pequenas unidades de processamento neural, ou *Neural Processing Unit* (NPU) diretamente nos dispositivos, fornecendo meios de processar as redes neurais em tempo real, localmente e com baixo consumo [MILLAR et al., 2025].

No trabalho de [YANG; SAKIYAMA; VERBAUWHEDE, 2006], foi implementado um sistema de captura de impressões digitais, extração de características e comparação de biometrias em tempo real, utilizando um processador *LEON-2*, embarcado em um *Field Programmable Gate Array (FPGA) Xilinx* e um sensor de captura de imagem para impressão digital. O processo completo da captura até a comparação da biometria é realizado em aproximadamente 4 segundos para um banco de 100 imagens.

Equipamentos comerciais, como o *Infineon - PSOC Fingerprint FPG1*[1], um microcontrolador com algoritmo de captura, extração de características e comparação de impressões digitais integrados. Este dispositivo realiza comparações em um banco interno de até 15 biometrias em um tempo de 375 milissegundos. O mesmo algoritmo pode ser utilizado em outros microcontroladores, por exemplo um *ARM CORTEX M4* rodando com clocks de 200 MHz, executando comparações de impressões em 250 milissegundos, com adição de 50 milissegundos por biometria cadastrada no banco de dados.

Equipamentos de baixo custo são amplamente utilizados para controle de acesso de prédios, universidades e residências, como o *iDAccess Bio*[2]. Este equipamento possui capacidade para 2000 impressões digitais no banco de dados interno, permitindo a expansão da quantidade de biometrias quando utilizado em conjunto com softwares de comparação instalados em computadores ou servidores. As comparações são realizadas nas biometrias

[1] Informações do equipamento disponível em : <https://www.infineon.com/products/microcontroller/32-bit-psoc-arm-cortex/fingerprint-m0-plus/fpg1#about>

[2] Informações do equipamento disponível em : <https://www.controlid.com.br/docs/idaccess-pt>

do banco de dados interno em poucos segundos, podendo levar algumas dezenas de segundos quando se utiliza bancos de dados e software externos.

1.1 Motivação

Equipamentos como relógios de ponto e controladores de acesso, que utilizam alguma tecnologia biométrica, possuem maior segurança e facilidade para os usuários. O uso dessa tecnologia dificulta que um usuário se passe por outro ou que esqueça seu dispositivo de acesso, como pode ocorrer ao utilizar cartões ou senhas.

Do ponto de vista operacional, quanto maior o tempo de processamento de um dado, maior será o custo financeiro e temporal para disponibilizar os recursos necessários para a execução da tarefa. O tempo gasto por um usuário pode ser crítico a ponto de inviabilizar a utilização de um sistema. Sistemas como relógios de ponto, que registram o horário de entrada e saída de um funcionário, podem sofrer atrasos devido ao processamento de dados ou pelas filas formadas por outros usuários aguardando o equipamento ficar livre. Esse tempo gasto pode impactar uma empresa com custos inesperados, como ações trabalhistas por problemas com a folha de ponto dos funcionários ou o custo envolvido quando um funcionário fica ocioso enquanto aguarda.

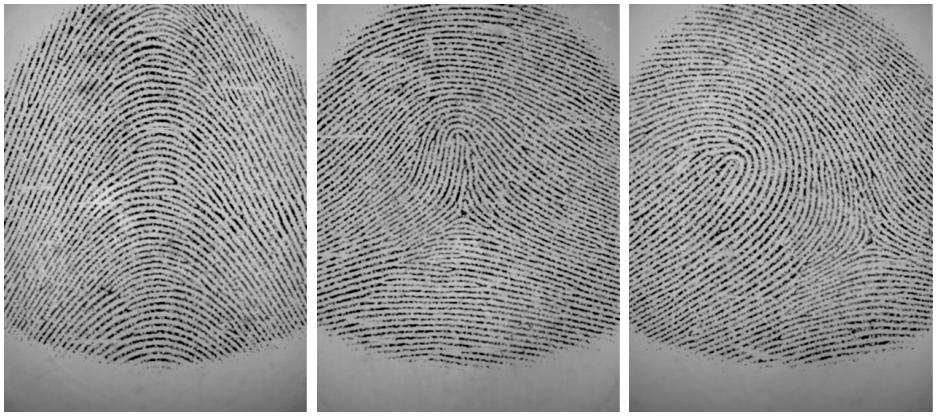
Espera-se que, com a aplicação de sistemas inteligentes de classificação, possa diminuir o tempo que o sistema gasta no total para identificar um usuário, reduzindo os custos operacionais.

1.2 Objetivos

Este estudo tem como objetivo principal, estudar e implementar algoritmos de classificação para reconhecimento de biometrias por impressão digital, de modo a reduzir os tempos de processamento de busca nos bancos de dados. Para tanto, serão empregados sistemas baseados em redes neurais com aprendizado profundo.

As biometrias foram classificadas em cinco grupos principais, conforme verificado na Figura 4, sendo estes os grupos *arch* (A), *tented arch* (TA), *left loop* (LL), *right loop*(RL) e *whorl*(W), ou arco, arco agudo, laço esquerdo, laço direito e redemoinho, em tradução livre. Estas classes não possuem uma distribuição uniforme entre a população amostrada, apresentando uma estimativa de 3,7%, 2,9%, 33,8%, 31,7% e 27,9%, respectivamente [MALTONI, 2004].

A busca de usuários, idealmente, seria realizada no formato 1:1, de modo que o mecanismo de busca possui as informações do usuário e realiza a autenticação por meio da verificação da correspondência destas com os dados biométricos adquiridos, conforme ilustrado na Figura 5. Em sistemas de controle de acesso, ou dispositivos de baixo custo,



(a) *Arch* (b) *Tented Arch* (c) *Left Loop*

(d) *Right Loop* (e) *Whorl*

Figura 4 – Diferentes padrões de biometrias. Fonte: Autor

as buscas de usuários geralmente são realizadas no formato 1:N, de modo que o mecanismo de busca desconhece as informações do usuário, sendo necessário verificá-lo em todo o banco de dados,conforme a Figura 6. Esse tipo de busca utilizará no pior caso N iterações de validação do usuário. Portanto, espera-se que uma melhor divisão nos bancos de dados, permitida pela classificação, reduza consideravelmente o número de iterações e tempo de busca [CORMEN et al., 2009].

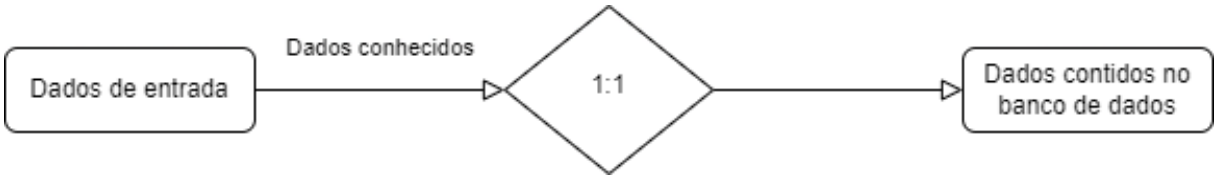


Figura 5 – Estrutura de busca no formato 1:1. Fonte: Autor

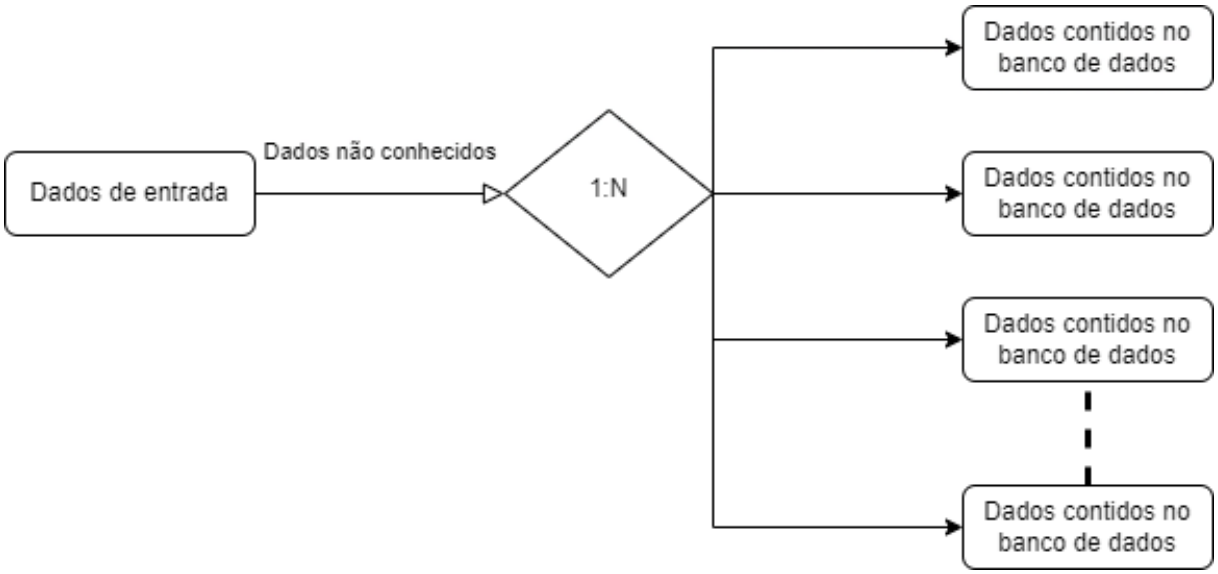


Figura 6 – Estrutura de busca no formato 1:N. Fonte: Autor

2 Fundamentação Teórica

As propriedades comuns de biometrias têm sido usadas para comparação e classificação a partir de características físicas individuais de uma pessoa, como rosto, íris e impressão digital. Como os atributos das impressões digitais são únicos e raramente alterados, esse tipo de biometria tornou-se um dos principais métodos para aplicações em sistemas. O objetivo desses sistemas é buscar uma detecção e classificação robusta e rápida. Os algoritmos realizam análise de recursos e correspondência usando informações de impressão digital para identificação pessoal, exigindo um enorme número de processos de comparação com um banco de dados de impressões digitais geralmente grande. Portanto, os sistemas geralmente não conseguem retornar o resultado em tempo real. Fundamentalmente, os algoritmos categorizam as imagens de impressão digital em três níveis: padrão (fluxo e orientação da crista), pontos (terminações de crista por fricção) e forma (borda, largura e cicatriz). Assim, a alta qualidade e a discriminação das características desses três níveis são consideradas para obter as melhores características [RIM; KIM; HONG, 2021].

2.1 Métodos de captura de impressões digitais

Um dos aspectos mais importantes da identificação de biometrias são os sensores, sendo estes os responsáveis por adquirir atributos biométricos fisiológicos ou comportamentais do usuário e convertê-los em sinais digitais, que podem ser utilizados pelo sistema. Atualmente, existem diversos tipos de sensores comerciais, cada qual com suas particularidades. Os tipos de sensores mais comuns são os sensores ópticos, capacitivos e de ultrassom.

2.1.1 Sensores ópticos

Ao pressionar o dedo em um prisma, os picos da correspondente impressão digital ficam em contato com a superfície do prisma. Por sua vez, os vales apresentarão uma camada de ar que distancia os vales da superfície, conforme a Figura 7. Uma fonte de luz é aplicada em um dos lados do prisma, sendo essa luz refletida pelos picos na superfície com uma intensidade maior do que nos vales. A luz refletida passa por uma lente que focaliza e filtra a luz para um *Charge Coupled Devices* (CCD) ou *Complementary Metal–Oxide–Semiconductor* (CMOS), ambos amplamente utilizados como sensores de imagens. As imagens capturadas,geralmente uma câmera CMOS de baixo custo, possuem tons de cinza de acordo com a quantidade de luz refletida pelos picos e vales [MALTONI et al., 2009].

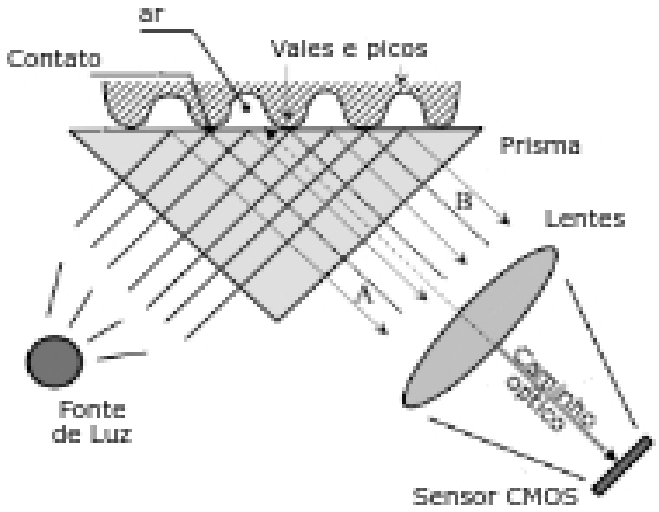


Figura 7 – Diagrama de um sensor óptico. Fonte: [MALTONI et al., 2009]

2.1.2 Sensores capacitivos

Estes sensores são geralmente compostos por matrizes de micro-capacitores ou pequenos sensores depositados pelo processo de *Thin-Film Transistor* (TFT), sobre uma camada de vidro ou plástico. Ao pressionar o dedo sobre a matriz de sensores, os picos da impressão digital ficam em contato com a superfície dos correspondentes elementos da matriz, de modo que a capacitância medida nessas posições é equivalente ao seu respectivo valor inicial. Em contrapartida, os vales apresentam uma camada de ar que distancia os vales da superfície. Assim, a capacitância do vale será equivalente à capacitância do ar e à capacitância da superfície em série, conforme exemplificado na Figura 8. Dessa forma, a imagem da biometria é gerada pela verificação das variações das capacitâncias medidas pelos sensores que compõem a matriz e suas respectivas posições [MALTONI et al., 2009].

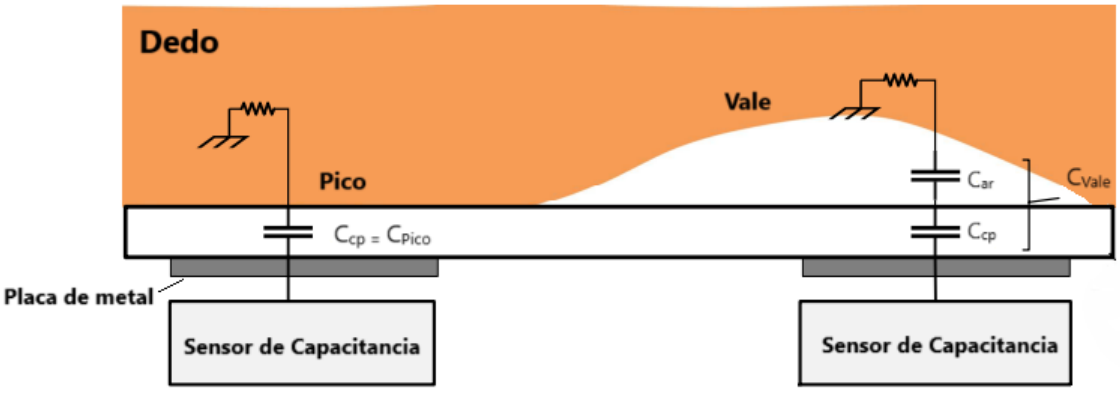


Figura 8 – Diagrama de um sensor capacitivo. Fonte: [HASSAN; KIM, 2018]

2.1.3 Captura por ultrassom

Estes sensores são geralmente compostos por matrizes de transdutores de ultrassom, ou seja, pequenos transmissores e receptores de sinais acústicos, instalados sob uma camada de vidro ou plástico, de modo que, esses materiais possuem uma impedância acústica semelhante à impedância da pele e da água. Os vales de uma impressão digital, possuirão uma camada de ar entre a epiderme e o meio condutor da onda, gerando um descasamento de impedância. Os picos, por sua vez, interfaceiam o sensor e por possuírem uma impedância semelhante ao material, permitem a propagação da onda com baixas perdas. Os transdutores transmitem as ondas ultrassônicas e aguardam a reflexão das ondas propagadas. As ondas refletidas nos vales, perderão pouca ou nenhuma energia, devido à grande diferença de impedância do meio em relação ao ar. Os picos, por permitirem a continuidade da propagação da onda acústica, refletem parte do sinal na camada da epiderme e parte do sinal na camada da derme. Por meio dessas variações de reflexões, os receptores são capazes de diferenciar os picos e vales nas digitais, bem como a diferenciação das biometrias da derme ou da epiderme, conforme demonstrado na Figura 9. As biometrias da derme e epiderme são correlatas, de modo que, a absência da biometria da derme pode indicar a utilização de uma impressão digital falsa [TANG et al., 2016].

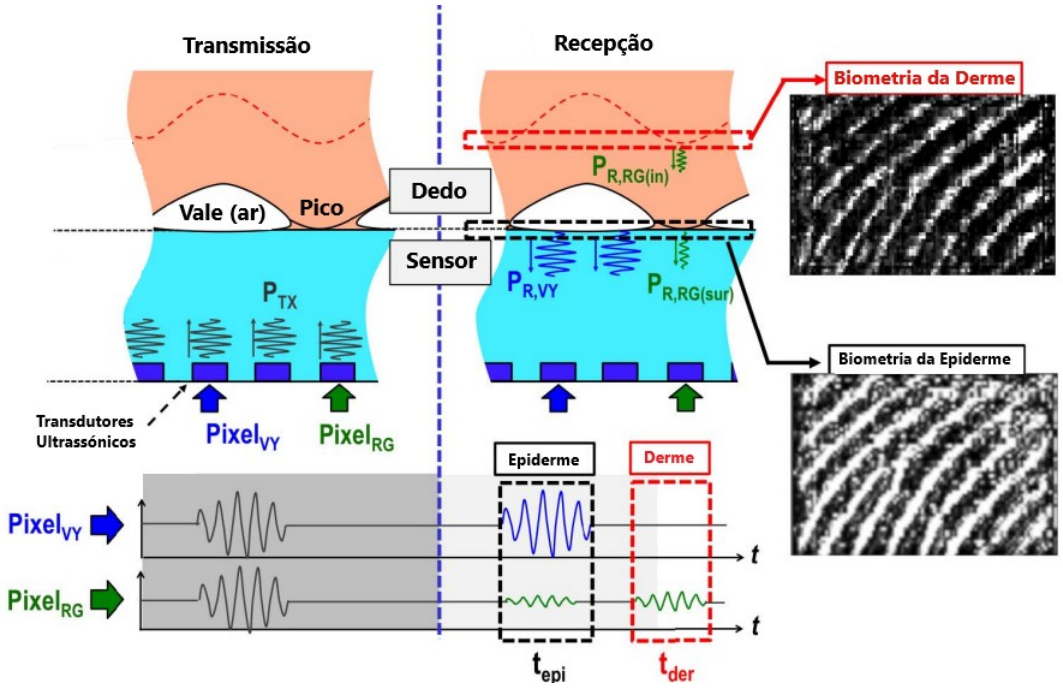


Figura 9 – Diagrama de um sensor Ultrassônico. Fonte: [TANG et al., 2016]

2.2 Fundamentos de redes neurais

Redes neurais artificiais, *Artificial Neural Networks (ANN)* em inglês, são técnicas de aprendizado de máquina inspiradas no mecanismo de aprendizado de organismos

biológicos. Os neurônios, células do sistema nervoso, são conectados entre si por meio dos axônios e dendritos, de modo que essas conexões são chamadas de sinapses, que constituem estruturas especializadas que permitem a transferência eficiente e direcionada de impulsos nervosos de uma célula para outra [KANDEL, 2000].

O termo "redes neurais", *Neural Network (NN)* em inglês, utilizado neste trabalho, refere-se às redes neurais artificiais em vez das biológicas. As NNs são unidades computacionais conectadas entre si por meio de "pesos", que são análogos às forças das conexões das sinapses nos organismos biológicos. Cada entrada de um neurônio é multiplicada por um peso, influenciando o resultado da função calculada pela unidade computacional. Assim, a informação inserida na entrada é propagada por meio de uma soma ponderada, que constitui a entrada liquida da função de ativação, a qual é então utilizada para se gerar a saída do neurônio. A Equação 2.1, expressa a soma ponderada, onde N é o número de entradas, x_i a i-ésima entrada do neurônio, w_i o peso associado à i-ésima entrada, b um viés, resultando na entrada liquida da função de ativação u. Essa função é inspirada no mecanismo de disparo de um neurônio biológico e determina qual valor de saída deve ser transmitido ao próximo neurônio, de modo que a saída final do neurônio (y) é o resultado da aplicação da função de ativação ($\phi(u)$), conforme a Equação 2.2. A função ativação, pode assumir funções como degrau unitário, logística (sigmóide), Tangente Hiperbólica (Tanh), entre outros. A Figura 10 mostra um neurônio artificial e de um neurônio biológico.

$$u = b + \sum_{i=1}^{N} w_i x_i \tag{2.1}$$

$$y = \phi(u) = \phi\left(b + \sum_{i=1}^{N} w_i x_i\right) \tag{2.2}$$

A aprendizagem do sistema ocorre por meio da alteração dos pesos que conectam os neurônios de modo a minimizar o valor de uma função custo que, no caso do aprendizado supervisionado, é utilizada para avaliar a discrepância entre a saída da rede e valores de treinamento conhecidos, como o erro quadrático médio e a entropia cruzada. Cabe destacar que, assim como um estímulo externo é necessário para o aprendizado nos organismos biológicos, o análogo ao estímulo externo em uma NN é dado por um conjunto de dados de treinamento que contêm exemplos de pares entrada-saída da tarefa a ser aprendida. Por exemplo, os dados de treinamento podem conter representações de imagens (entrada) e suas respectivas classes, previamente anotadas (saída). Os pares entrada-saída dos dados de treinamento são alimentados na NN para predizer a categoria ou nome da classe. Os pesos por sua vez são ajustados na NN em resposta aos erros de predição do sistema por meio do algoritmo de retropropagação. O intuito de alterar os pesos é de modificar os cálculos da rede com a finalidade de melhorar o desempenho das predições realizadas pela NN em cada iteração de treinamento. Sendo assim, os pesos são atualizados com a

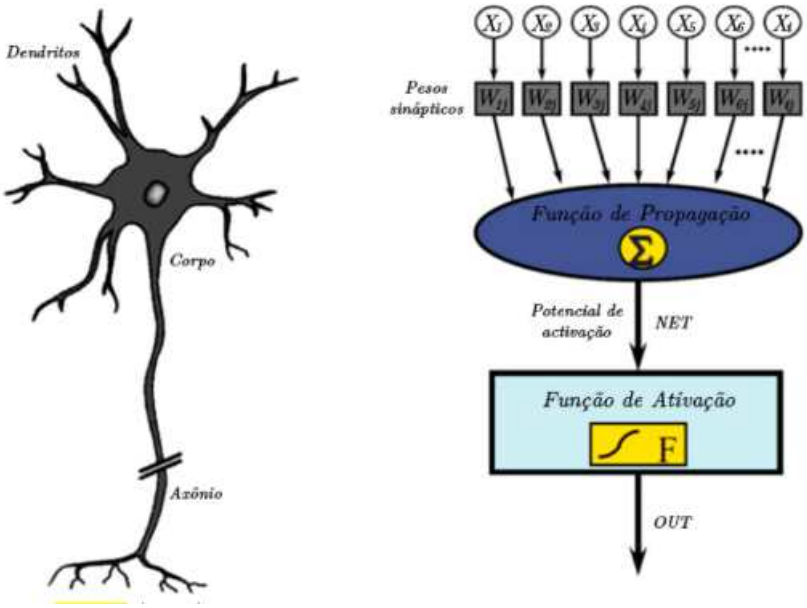


Figura 10 – Representação de um neurônio artificial e biológico. Fonte: [NARANJO, 2014]

intenção de reduzir uma função de perda calculada em função dos pares entrada-saída que compõem o conjunto de dados de treino. Desse modo, por exemplo, se uma rede é treinada utilizando diferentes imagens de cães, o sistema será útil se, eventualmente for capaz de reconhecer um cão em uma imagem nunca apresentada antes. Essa capacidade de realizar predições corretamente a partir de entradas não conhecidas anteriormente a partir de um conjunto finito de pares de treinamento é chamada de generalização. Desse modo a função primária de um modelo de aprendizado de máquina é a formar um sistema capaz de generalizar seu aprendizado dos dados de treinamento, mantendo desempenho compatível ao ser exposto a exemplos novos previamente desconhecidos [AGGARWAL et al., 2018].

Uma rede neural passa a ser chamada de rede neural profunda, *Deep Neural Network* (DNN) em inglês, quando há diversas camadas ocultas entre a camada de entrada e a camada de saída. A Figura 11 mostras dois exemplos, salientando a diferença entre os tipos de redes.

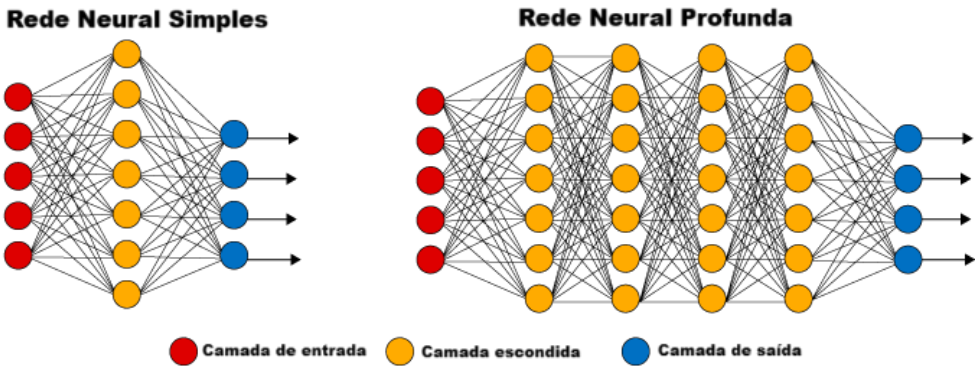


Figura 11 – Comparação entre uma NN e uma DNN. Fonte: [BOOK, 2023]

2.2.1 Redes Neurais Convolucionais

As redes neurais convolucionais, *Convolutional Neural Networks (CNN)*, são redes neurais profundas amplamente utilizadas em visão computacional para tarefas de detecção, reconhecimento e classificação de objetos.

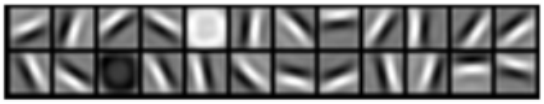
Uma típica camada convolucional, consiste de três estágios. O primeiro estágio aplica as convoluções em paralelo para gerar uma série de ativações lineares. O segundo estágio aplica a cada uma destas ativações lineares uma função não linear, consistindo no estágio de detecção. No terceiro estágio é realizada uma função de agrupamento, ou *pooling* em inglês, em que um dos objetivos é diminuir a dimensionalidade da saída da camada [GOODFELLOW; BENGIO; COURVILLE, 2016]. A utilização de diversas camadas convolucionais permite o aprendizado e a extração de características da imagem de entrada.

A razão para isso é que diferentemente das DNNs, onde cada neurônio se conecta a todos os neurônios da próxima camada (conexão completa), as CNNs utilizam conexões esparsas. Em uma CNN, um neurônio da camada seguinte se conecta apenas a um subconjunto de saídas da camada anterior. Desse modo, essa arquitetura permite que os neurônios agrupem características locais usando o mesmo filtro, resultando em um mapa de características, *feature map* em inglês, onde cada mapa representa a resposta a um filtro específico aprendido em toda a imagem [VARGAS; PAES; VASCONCELOS, 2016].

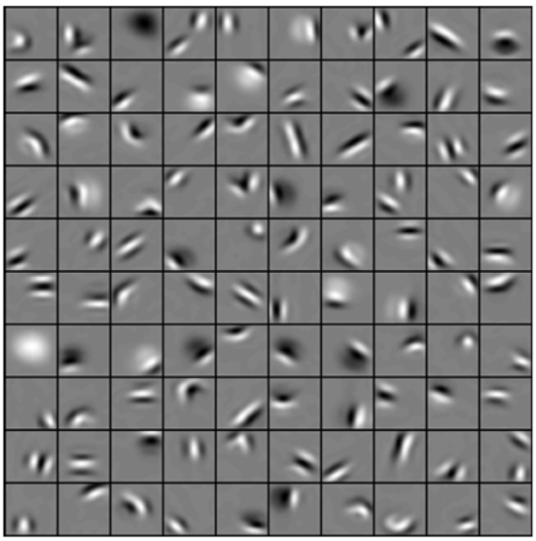
Diferentes algoritmos, ou diferentes arquiteturas de NN, são obtidos por diferentes arranjos de camadas, ordenados de modo a propiciar o aprendizado de um conjunto de características especializadas para a tarefa para qual ela foi treinada. Em camadas convolutivas mais próximas da entrada, os filtros geralmente possuem mapas de características mais simples, como pontos e traços em posições específicas. Nas camadas seguintes os filtros podem conter formas a padrões mais complexos em regiões de maior importância, por exemplo pedaços ou formas de uma face humana [LEE et al., 2009], como mostrado na Figura 12.

As ultimas camadas são, por via de regra, totalmente conectadas ou *Fully Connected* (FC) em inglês. Essas, combinam as respostas das anteriores sendo, de grosso modo, responsáveis pela classificação propriamente dita. Após a última camada totalmente conectada, finalmente obtemos a camada de saída da CNN estruturada, geralmente por uma função que fornece a probabilidade da resposta pertencer a um dos grupos de treinamento [VARGAS; PAES; VASCONCELOS, 2016].

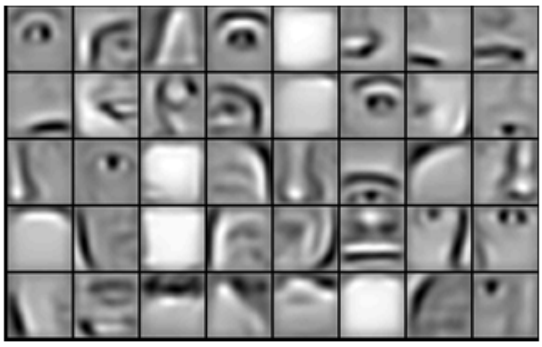
A Figura 13 ilustra a divisão entre as camadas que predominantemente são destinadas à extração de características e as que são predominantemente responsáveis pela classificação. Este modelo conta com três camadas de convolução e agrupamento, seguidas de duas camadas totalmente conectadas e, finalmente, a camada de saída, em que as saídas



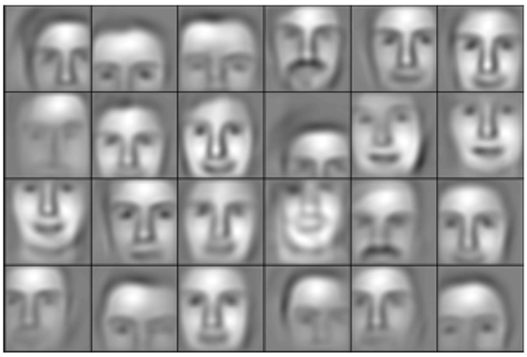
(a) Filtros simples.



(b) Filtros de formas em regiões de maior importância.



(c) Filtros semelhantes às partes de uma face.



(d) Partes ou faces de uma pessoa.

Figura 12 – Criação de filtros pelas camadas de convolução. Fonte: [LEE et al., 2009]

correspondem a estimativas de probabilidade de entrada pertencer a cada classe.

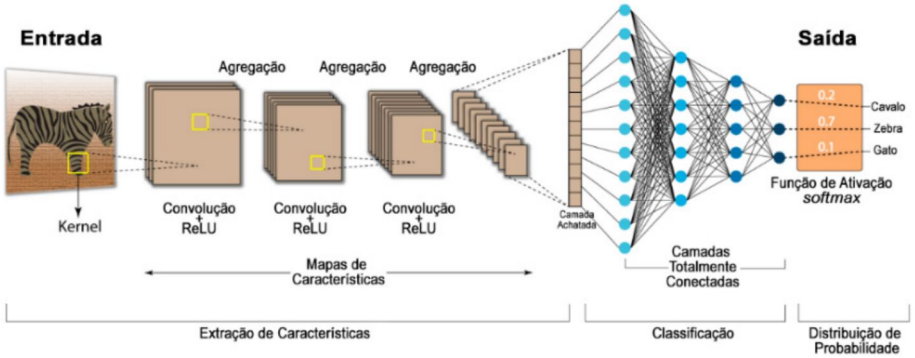


Figura 13 – Uma CNN de classificação e reconhecimento de imagens. Fonte: [FERREIRA et al., 2023]

2.2.1.1 Convolução

De forma geral, a convolução é uma operação entre duas funções com argumentos de valores reais, gerando uma terceira conforme Equação 2.3 para o caso continuo e unidimensional, em que esta operação é denotada por um asterisco.

$$(K * I)(t) = \int_{-\infty}^{\infty} K(\tau) I(t - \tau)\, d\tau \tag{2.3}$$

No conceito de *CNNs*, a operação de convolução geralmente é aplicada em sistemas com matrizes multidimensionais, podendo ser descrita pela Equação 2.4, para o caso discreto e bidimensional, em que o primeiro argumento (a função K que refere-se ao *kernel* e tem dimensões m x n) e o segundo argumento (a função I que refere-se à entrada e tem dimensões i x j), geram uma saída, a *feature map* [GOODFELLOW; BENGIO; COURVILLE, 2016].

$$S(i, j) = (K * I)(i, j) = \sum_{m} \sum_{n} I(i - m, j - n) K(m, n) \tag{2.4}$$

Em uma camada de convolução, os pesos respectivos de cada uma das conexões entre os neurônios, podem ser interpretados como uma matriz que representa o filtro de uma convolução de imagens no domínio espacial (também conhecido como *kernel* ou máscara). Cada *kernel* produz uma saída propagada para a camada seguinte, sendo que cada camada de convolução é composta por diversos neurônios, cada qual responsável por aplicar uma máscara em um pedaço específico da imagem. A saída de uma camada se conecta com a próxima camada através dos pesos associados à resposta impulsiva do *kernel*. Durante o treinamento do modelo, ocorre um compartilhamento dos pesos das conexões neurais, permitindo que um mesmo *kernel* possa ser aplicado em diversas posições de uma mesma imagem ou matriz. Essa reutilização de pesos permite reduzir o número de parâmetros e tempo de treinamento do modelo [VARGAS; PAES; VASCONCELOS, 2016]. A Figura 14 exemplifica a operação de convolução de uma matriz de 3x4, por um *kernel* de matriz 2x2, gerando uma saída de matriz 2x3.

O compartilhamento de pesos diminui significativamente o número de parâmetros a serem aprendidos e o tempo de treinamento da rede, consequentemente. De maneira a simplificar a esquematização dessas redes, um filtro a ser aplicado na imagem é representado visualmente por apenas um neurônio, trazendo a falsa sensação de que ele faria sozinho o processo de percorrimento da imagem.

O passo, ou *stride*, define quantos pixels serão saltados para a realização da próxima operação de convolução, consequentemente, definindo o tamanho da saída da operação de convolução. Figura 15 demonstra uma operação de convolução com variação do *stride*.

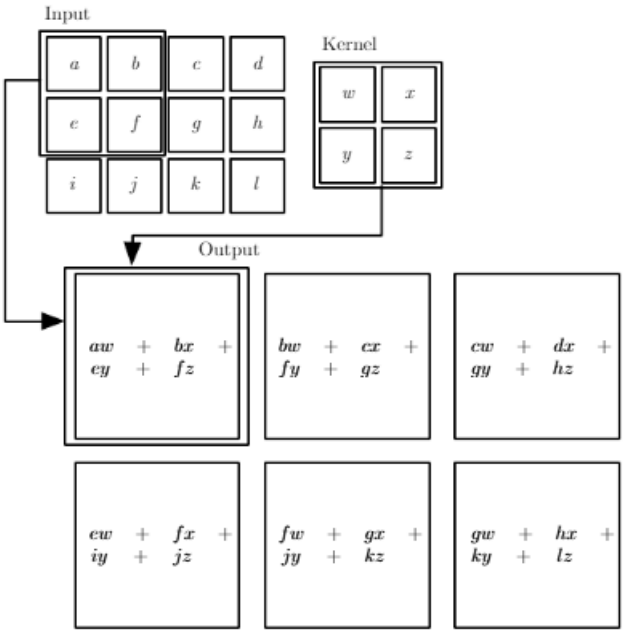


Figura 14 – Exemplo de uma operação de convolução. Fonte: [GOODFELLOW; BENGIO; COURVILLE, 2016]

Diferentemente do processo tradicional de visão computacional, no qual o modelo inicial parte da definição manual de filtros ou *features* a serem utilizadas, nas camadas de convolução das CNNs é preciso definir somente a quantidade de filtros e o tamanho do *kernel*, bem como o *stride*, ou passo de convolução, como mostrado na Figura 15 aplicado ao *kernel* por camada. Não é indicado ao sistema em nenhum momento prévio, qual filtro deve ser utilizado. O processo de aprendizado da rede altera os pesos ao longo do treinamento, até se obter os melhores valores dos pesos dos filtros para o conjunto de dados utilizado [VARGAS; PAES; VASCONCELOS, 2016].

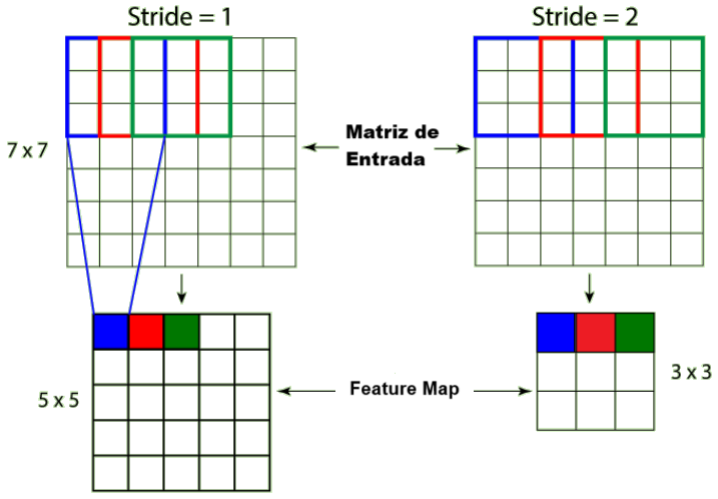


Figura 15 – Exemplo de convolução com variação de *stride*. Fonte: [KATEGARU, 2020]

2.2.1.2 Função de ativação

Logo após a transformação dos dados em uma camada convolucional de uma CNN, é aplicada o equivalente à uma função de ativação, que a via de regra é não linear, formando a camada de detecção [GOODFELLOW; BENGIO; COURVILLE, 2016].

Um exemplo de uma função de ativação não linear, é a unidade retificada linear, ou *Rectified Linear Unit* (*ReLu*) em inglês. A *ReLu* converte valores negativos em valores nulos, mantendo os valores positivos no mapa de características resultante [ANDRADE, 2022], conforme demonstrado na Figura 16.

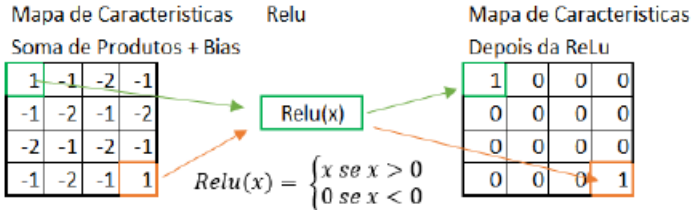


Figura 16 – Aplicação da ativação *ReLu* em um *feature map*. Fonte: [ANDRADE, 2022]

Em camadas de saída com no mínimo três classes mutuamente exclusivas, geralmente é utilizada a função *SoftMax*, também conhecida como função exponencial normalizada, é usada para converter um vetor de K números reais (chamados logits ou pontuações) em uma distribuição de probabilidade de K resultados possíveis. A saída $\sigma(z_i)$ para o i-ésimo elemento do vetor de entrada z é dada pela Equação 2.5. De forma geral, a função transforma a ultima camada FC em uma estimativa de distribuição de probabilidade com somatória igual a um. Essa transformação permite uma interpretação do resultado e permite a realização da predição pela escolha da classe com a maior probabilidade estimada, como exemplificada na camada de saída da Figura 13.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.5}$$

2.2.1.3 Agrupamento

Outra camada muito importante comumente utilizada após as camadas de convolução e detecção é a camada de agrupamento, ou *pooling* em inglês. A função dessa camada é reduzir a dimensionalidade dos dado na rede. Essa redução é importante não só por questão de agilidade no treinamento, mas principalmente para criar invariância espacial. A camada de *pooling* funciona principalmente de duas formas, média e máximo, ou *Average* e *Max* em inglês, dos elementos de uma região de *feature map* produzido pela camada de detecção.

A Figura 17 mostra o agrupamento do conjunto de dados, neste exemplo uma matriz ou imagem de 4x4 é agrupada em blocos menores de acordo com o *kernel* de 2x2

e passo (*stride*) de 2, de modo que a matriz resultante da esquerda agrupa os valores máximos (*MaxPooling*) do *kernel*, já a matriz na direita agrupa os valores em uma média (*AveragePooling*) entre os valores do *kernel*, ambos os agrupamentos criam uma redução da dimensão para 2x2 [VARGAS; PAES; VASCONCELOS, 2016].

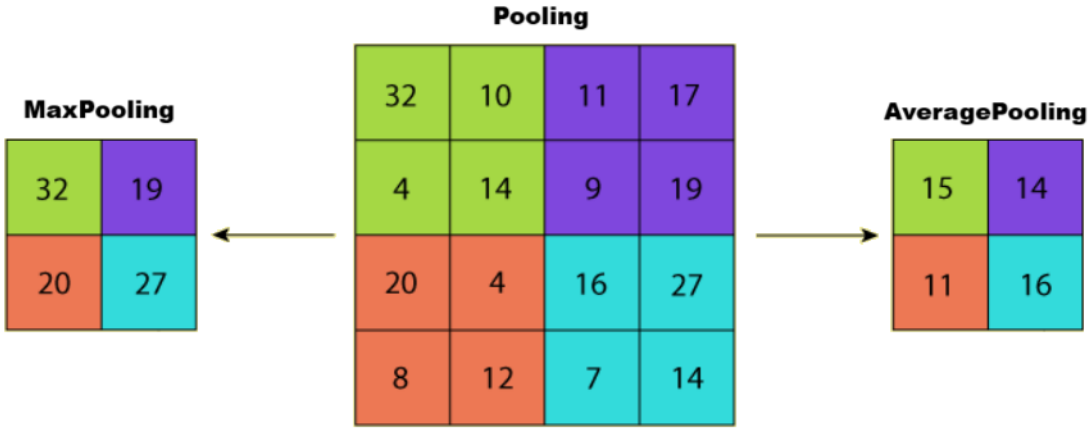


Figura 17 – Exemplo de uma operação de *MaxPooling* na esquerda e *AveragePooling* na direita. Fonte: [KATEGARU, 2020]

2.2.1.4 Camada totalmente conectada

As camadas totalmente conectadas, ou *Fully Connected (FC)* em inglês, possuem esta denominação pois a saída de todos os neurônios de uma camada se conectam aos neurônios da próxima camada. A camada recebe os mapas de características da ultima camada convolucional, obtidos durante o treinamento do modelo, após a operação de achatamento (*flatten*) para um vetor, permitindo que as entradas da camada FC sejam conectadas a todos os neurônios da camada. As últimas camadas da Figura 13 formam um exemplo de aplicação de camadas FC [ANDRADE, 2022].

2.2.1.5 Normalização

As camadas de normalização têm função de estabilizar a distribuição das ativações dentro de uma DNN. Essa estabilização se faz necessária porque, durante o treinamento por retropropagação, as atualizações dos parâmetros das camadas iniciais modificam continuamente a distribuição das entradas para as camadas subsequentes, fenômeno conhecido como Mudança de Covariância Interna, ou *Internal Covariate Shift* em inglês [IOFFE; SZEGEDY, 2015].

Em NNs, os gradientes podem crescem exponencialmente durante a retropropagação, fenômeno chamado de gradiente explosivo, o que resulta em atualizações de peso descontroladas, treinamento instáveis e modelos não convergentes. Em contrapartida, quando os gradientes possuem crescimentos pequenos, chamado de gradiente evanescente, resulta em gradientes que tendem a zero nas camadas iniciais. Isso impede o aprendizado

efetivo dos pesos mais próximos da entrada. Ao fixar as estatísticas das entradas de cada camada, a normalização garante que os gradientes permaneçam em uma faixa de valores adequada. Evitando a saturação dos neurônios, prevenindo a explosão ou o desaparecimento dos gradientes, assim permitindo que o modelo seja treinado de forma mais rápida e estável [GOODFELLOW; BENGIO; COURVILLE, 2016].

O modelo AlexNet [KRIZHEVSKY; SUTSKEVER; HINTON, 2012] foi um dos primeiros modelos de CNN a utilizar normalizações de resposta local, ou *Local Response Normalization* (LRN) em inglês. Este mecanismo faz com que neurônios com uma alta ativação inibam a ativação dos neurônios vizinhos, criando um tipo de contraste local que ajuda a realçar as características mais importantes e a suprimir as menos relevantes.

As camadas de normalização em lotes, ou *batch normalization* em inglês, são mecanismos que calculam a média e o desvio padrão de um mini-lote de dados durante o treinamento. Em seguida, utilizam-se esses valores para normalizar as ativações, garantindo que a média seja zero e o desvio padrão seja unitário, melhorando o desempenho e a estabilidade do modelo [IOFFE; SZEGEDY, 2015]. Esta técnica de normalização gera uma resposta em todo o lote de treino, diferentemente da técnica LRN que gera uma normalização local em um pequeno subconjunto de ativações da camada.

2.2.1.6 Dropout

Em modelos muito grandes, pode ocorrer uma falta de generalização, onde os dados de treinamento criam filtros com características especializadas apenas para os dados de treinamento, possivelmente negligenciando outros filtros de características mais genérica e mais relevantes. Dessa forma, aplica-se a camada de *Dropout*, que consiste, tipicamente, em desativar aleatoriamente neurônios das camadas totalmente conectadas de NNs muito grandes, permitindo que o modelo seja regularizado durante o treinamento. A cada iteração de treinamento, pode-se interpretar que são formadas sub-redes que compartilham um conjunto de pesos a partir da desativação de neurônios da rede original. Além disso, há o fato de que seria computacionalmente custoso treinar os modelos com todos os neurônios disponíveis, portanto este procedimento permite que um número exponencial de sub-redes sejam consideradas ao mesmo tempo que uma quantidade menor de memória é necessária durante o treinamento [GOODFELLOW; BENGIO; COURVILLE, 2016].

A Figura 18 exemplifica a desativação de neurônios aleatórios a partir de uma rede neural de pequeno porte.

2.3 Algoritmo de busca

Os algoritmos de comparação de biometrias, de forma genérica, capturam as características de uma biometria de entrada e realizam uma busca para determinar se

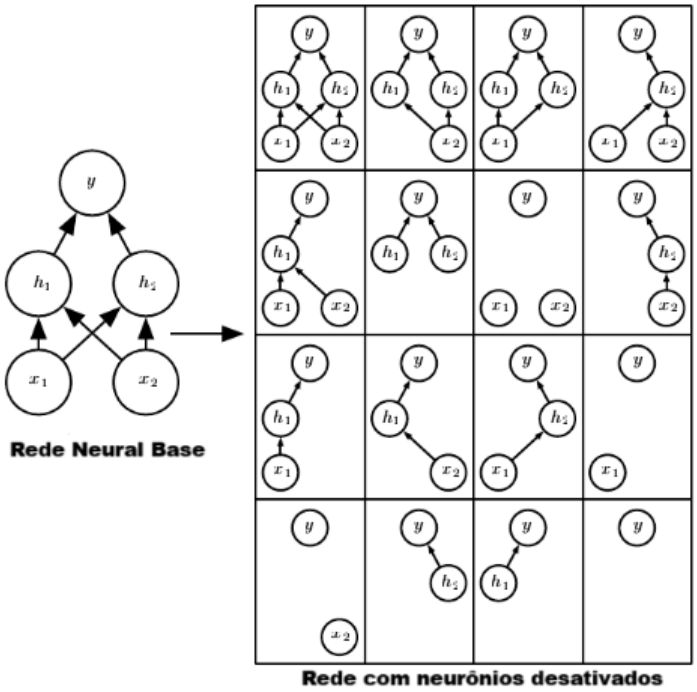


Figura 18 – Exemplo de desativação de neurônios por Dropout de uma rede neural. Fonte: [GOODFELLOW; BENGIO; COURVILLE, 2016]

a mesma corresponde com alguma das imagens cadastradas no banco de dados. Em [MALTONI et al., 2009], são discutidos diversos algoritmos e métodos de comparação de biometrias digitais. Um dos mais utilizados é o método de minúcias, que transforma características, como as terminações de vales ou picos, em vetores com posição e ângulo. Em seguida, essas características são comparadas como uma máscara sobreposta sobre as imagens do banco, que podem ser rotacionadas ou deslocadas, como exemplificado na Figura 19.

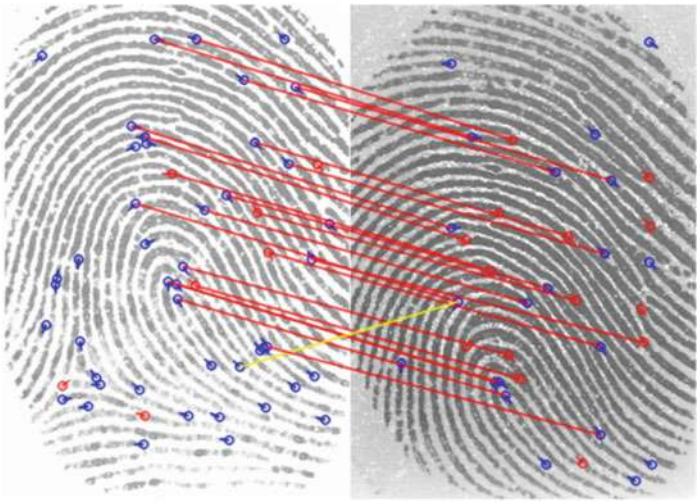


Figura 19 – Exemplo de comparação por minucias entre duas biometrias. Fonte: [MALTONI et al., 2009]

Neste trabalho, a comparação de biometrias assemelhar-se-á a uma comparação

simples entre dois textos quaisquer. Dessa forma, supõe-se que o sistema possuirá um tempo fixo de comparação para cada registro de biometria no banco de dados, de modo que esse tempo corresponderá a um valor fixo σ. Uma vez que o banco é sequencial, ou seja, as biometrias estão dispostas sequencialmente de acordo com a sua ordem de registro. O tempo de busca ($T_{Completo}$) depende da posição do registro no banco de dados (i_{global}), de acordo com a Equação 2.6.

$$T_{Completo} = i_{global} \times \sigma \tag{2.6}$$

Propõem-se dois algoritmos de busca, partindo da suposição de que a distribuição de biometrias possui uma estimativa não uniforme para os tipos *arch* com 3,7%, *tented arch* com 2,9%, *left loop* com 33,8%, *right loop* com 31,7% e *whorl* com 27,9%, de acordo com o verificado na obra de [MALTONI, 2004].

No primeiro caso, os algoritmos de classificação do tipo de impressão digital são utilizados apenas durante o cadastro da biometria do usuário. Dessa forma, o banco de dados será composto por cinco classes de dados, com distribuição semelhante à descrita acima. Dessa forma, a busca e comparação de uma nova biometria são realizadas prioritariamente na classe de maior probabilidade e , sequencialmente, nas outras classes em ordem decrescente de tamanho. Assim, o tempo de busca total ($T_{Algoritmo}$) é determinado pela soma do tempo gasto na busca e comparação do registro em cada classe na base de dados, de acordo com a ordem dada pela distribuição probabilística estimada das classes. A Equação 2.7 quantifica o tempo, dado que Cl diz respeito às classes e N_{Cl} é a quantidade de registros dentro desta classe. Neste algoritmo $Cl = 1$ corresponde a classe com maior distribuição de biometrias (*left loop*), sendo os demais valores de Cl correspondentes as demais classes em ordem decrescente de tamanho.

$$T_{Algoritmo} = \sum_{Cl=1}^{6} \sum_{i_{Cl}=1}^{N_{Cl}} (i_{Cl} \times \sigma) \tag{2.7}$$

No segundo algoritmo de busca proposto, a classificação é realizada durante o cadastro e a cada nova requisição de comparação. Dessa forma, a base de dados é particionada entre as cinco classes. A busca e comparação de uma nova biometria, são realizadas prioritariamente nas classes de maior probabilidade dadas pelo algoritmo de classificação e sequencialmente nas outras classes. Neste caso, o tempo de busca total ($T_{Algoritmo}$) se dá pela somatória de tempo gasto na busca e comparação do registro em cada classe na base de dados, de acordo com a ordem dada pelo algoritmo de classificação. Sendo assim a Equação 2.7 quantifica o tempo, dado que o valor de Cl identifica uma classe e N_{Cl} é a quantidade de registros dentro desta classe. Neste algoritmo, $Cl = 1$ corresponde à classe no qual o algoritmo de predição resultou na maior probabilidade de classificação, sendo os

demais valores de Cl correspondentes às demais classes em ordem crescente do tamanho das probabilidades.

Em todos os casos, sendo a busca no banco de dados sem divisões ou com as duas opções de algoritmos propostos, o tempo máximo de busca será dado pela Equação 2.6 onde o $i_{global} = N_{Total}$ e N_{Total} é a quantidade total de biometrias no banco.

3 Dataset

O *dataset* utilizado neste trabalho pode ser dividido em duas partes principais, uma contendo partes do banco de dados *Casia-FingerprintV5*[1] e outra através do software *Syntetic Fingerprint Generator* (SFinGe)[2], capaz de gerar imagens de impressões digitais artificiais.

O banco de dados de impressões digitais (*Casia-FingerprintV5*) foi criado pela Academia Chinesa de Ciências, no Instituto de Automação (CASIA), contendo em sua totalidade 20.000 imagens de biometrias, capturadas de 500 voluntários. A captura utilizou o sensor óptico da fabricante Digital Persona, modelo URU4000, mostrado na Figura 20. Cada voluntário contribuiu com 40 imagens, sendo 5 imagens por dedo para cada um dos 8 dedos, excluindo-se os dedos mínimos de cada mão. As imagens capturadas possuem resolução de 328x356 e gradação em escala de cinza de 8 bits.

Figura 20 – Sensor de captura URU4000.

As impressões digitais da base de dados *Casia-FingerprintV5* não possuem classificação por tipo de biometria, apenas identificação do número de usuário, número do dedo e número da imagem deste dedo. Sendo assim, as biometrias foram divididas manualmente em suas respectivas classes. As imagens possuem capturas de um mesmo dedo, com diferentes níveis de rotação e pressão no sensor de captura, de modo que ambos os fatores impactam diretamente o posicionamento, a qualidade e a visibilidade das características necessárias para a validação e classificação das imagens. A Figura 21 mostra o exemplo de um dedo capturado em diferentes posições e pressões no sensor de captura, evidenciando a dificuldade de distinguir visualmente o tipo de impressão digital deste dedo. Portanto, parte das imagens foi descartada devido a dificuldade de classificação individual das imagens. Após o descarte, restaram aproximadamente três mil biometrias deste banco de dados,

[1] Banco de dados disponível em : <http://english.ia.cas.cn/db/201611/t20161101_169922.html>
[2] Software disponível em : <https://biolab.csr.unibo.it/ResearchPages/SFinGe_Download.asp>

porem algumas classes de biometrias digitais ficaram com uma quantidade de imagens muito abaixo da média das outras, criando desbalanceamento no dataset.

Figura 21 – Imagens de um mesmo dedo, na base de dados *Casia-FingerprintV5*. Fonte: Autor

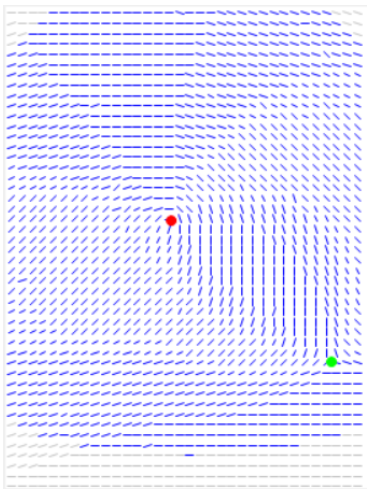
Para balancear a quantidade de biometrias entre as classes desejadas para a classificação, utilizou-se o software SFinGe. Na Figura 22 são mostrados os passos utilizados para a geração de uma impressão digital artificial, as imagens geradas possuem resolução de 416x560.

O software permite definir a identidade única do dedo sintético, baseando-se em dados de entrada como a classe de impressão digital e o posicionamento das singularidades. Por exemplo, em um *Left Loop*, o posicionamento de um *delta*, que são regiões onde três sistemas de picos se encontram (ponto verde), no lado direito e os núcleos, que são os pontos mais internos do padrão de picos em formato de laço, validam a estrutura fundamental da impressão, enquanto as minúcias como bifurcações e términos de picos em vermelho, enriquecem o detalhe único do padrão de maneira biologicamente plausível. A Figura 22a mostra os pontos de interesse e vetores de referência para a biometria pretendida. Em seguida, estes vetores geram as linhas características dos picos e vales, o software utiliza um filtro de Gabor, que se ajusta conforme a orientação e densidade dos vetores gerados. Este filtro é obtido pelo produto de uma gaussiana por uma função harmônica de cosseno. O resultado da aplicação deste filtro pode ser observado na Figura 22b. Pequenos cortes e imperfeições são adicionados na imagem, como pode ser verificado na Figura 22c. Na Figura 22d, ruído branco é adicionado para distorcer os padrões criados digitalmente, com o intuito de simular uma captura real. Em seguida, é aplicado um filtro no fundo da imagem, simulando um sensor de captura óptico ou capacitivo. Finalmente, a imagem é renderizada após todas as alterações realizadas pelos filtros e modificadores, observada na Figura 22f.

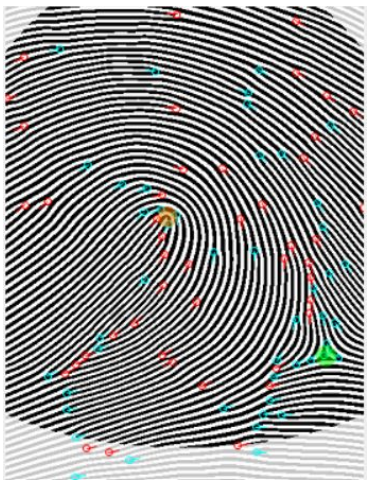
Este processo de geração foi realizado para cada uma das classes de impressões digitais estudadas. Dessa forma foram criadas aproximadamente 5000 impressões artificialmente, normalizando o *dataset* em aproximadamente 1400 imagens por classe, com

um total de 7215 biometrias no banco de dados. O Banco de dados foi então separado em cinco pastas, cada qual pertencente à classe de classificação desejada.

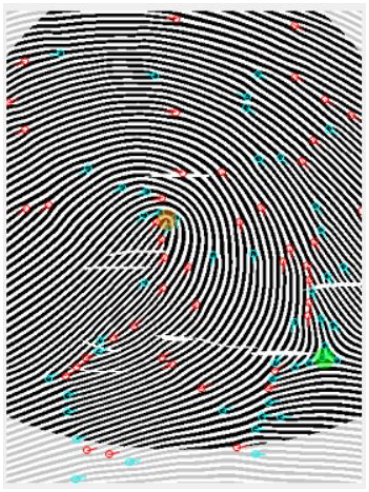
Espera-se que a equalização da quantidade de imagens por classe, melhore a generalização dos modelos e garanta que o gradiente de erro de todas as classes contribua de forma equitativa para a atualização dos pesos durante o treinamento.



(a) Criação de pontos e vetores.

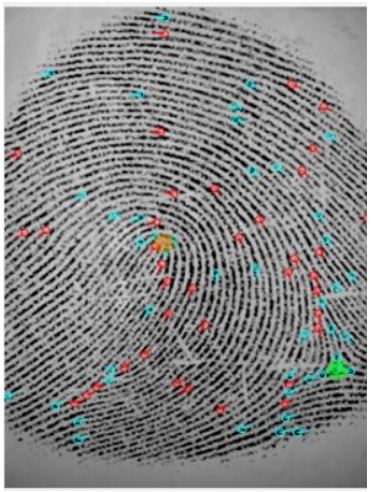


(b) Criação de picos e vales a partir dos vetores.



(c) Adição de cortes aleatórios.

(d) Adição de ruido na imagem.



(e) Filtro no fundo da imagem.

(f) Imagem da impressão digital finalizada.

Figura 22 – Criação de biometria artificial por meio do SFinGe. Fonte: Autor

4 Algoritmos de classificação

Os algoritmos de classificação utilizam *python*, aproveitando-se de estruturas e bibliotecas, principalmente da ferramenta Keras do *TensorFlow* (TF) para a leitura, treinamento e teste do código. O ecossistema TF é uma plataforma *open-source* abrangente, desenvolvida pelo Google, amplamente utilizada para a criação e treinamento de modelos de *Machine Learning*. Em sua forma completa, é otimizado para ambientes robustos de treinamento, como CPUs e GPUs em data centers ou estações de trabalho [ABADI et al., 2016]. Complementando o TF, temos o Keras, que atua como uma camada de aplicação de alto nível, destacando-se por sua interface intuitiva e modular, simplificando a definição de arquiteturas complexas e permitindo que a vasta maioria dos modelos de *Deep Learning* sejam construídos com agilidade [CHOLLET, 2017].

Uma vez que o modelo é construído e treinado no ambiente *TensorFlow*, a necessidade de implantá-lo em dispositivos com recursos limitados, como smartphones e equipamentos IoT, leva à adoção do *TensorFlow Lite* (TFLite)[1]. O TFLite é um conjunto de ferramentas que transforma o modelo para inferência diretamente no dispositivo, utilizando um formato otimizado e compacto. Esta otimização inclui técnicas como a quantização, que reduz a precisão dos dados do modelo (por exemplo, de 32 para 8 bits), minimizando drasticamente o tamanho do arquivo e acelerando o cálculo.

Neste trabalho, as execuções dos algoritmos propostos foram realizadas principalmente por meio da plataforma Google Colaboratory [BISONG et al., 2019].

O algoritmo do LeNet teve como aplicação mais importante a leitura automática de cheques utilizados pelo serviço postal norte americano e em diversas instituições bancárias, pavimentando o caminho para arquiteturas posteriores mais complexas. O modelo demonstrou a eficácia de uma rede neural que processa a entrada diretamente (pixels da imagem) sem a necessidade de uma fase de extração de características manual, sendo um dos primeiros algoritmos a aplicar o uso prático de CNNs [LECUN et al., 2002].

O modelo AlexNet, embora baseado na estrutura básica do LeNet, é significativamente maior e mais profundo, consistindo de oito camadas, cinco convolucionais seguidas por três camadas totalmente conectadas. Essa profundidade foi crucial para o sucesso em tarefas de reconhecimento de imagem complexas e em larga escala. As principais inovações técnicas do AlexNet incluíram o uso de múltiplas GPUs para acelerar o treinamento, a adoção da função de ativação ReLU, que se provou mais rápida no treinamento do que as funções sigmóide ou tanh, e a aplicação da técnica *dropout* nas camadas totalmente

[1] Informações sobre TensorFlow Lite estão disponíveis em : <https://www.tensorflow.org/lite/microcontrollers?hl=pt-br>

conectadas para mitigar o *overfitting*. A aplicação do modelo comprovou o poder das CNNs em processar dados brutos de imagem, ultrapassando os métodos tradicionais de engenharia de recursos e estabelecendo um novo paradigma para a pesquisa em Visão Computacional [KRIZHEVSKY; SUTSKEVER; HINTON, 2012].

Os algoritmos LeNet e AlexNet foram estruturados para se assemelhar ao apresentado por [WU; ZHU; GUO, 2020]. Os algoritmos completos podem ser verificados no Apêndice A e Apêndice B, respectivamente. A Figura 23 e a Figura 24 resumem as arquiteturas desses modelos aplicados para à classificação pretendida.

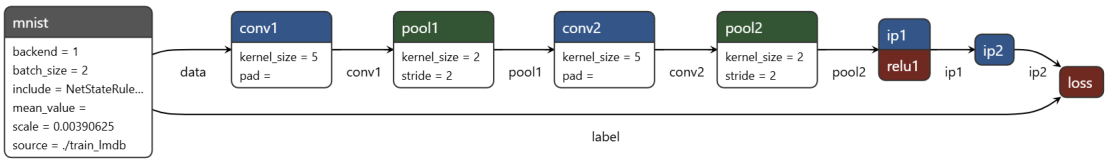


Figura 23 – Modelo de classificação LeNet. Fonte: Autor

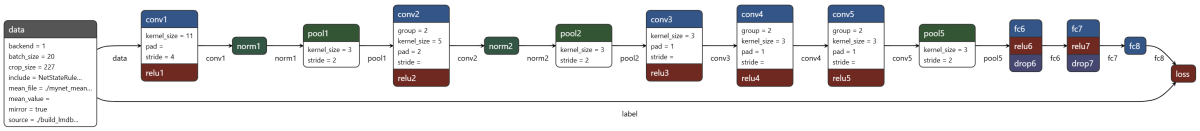


Figura 24 – Modelo de classificação AlexNet. Fonte: Autor

O modelo CaffeNet é significativo por ser uma reimplementação da arquitetura do AlexNet, utilizando o *framework Caffe*. A principal contribuição do CaffeNet foi provar que os resultados de alta precisão obtidos pelo AlexNet em tarefas de reconhecimento visual de grande escala poderiam ser reproduzidos de forma eficiente por um *framework* de código aberto e orientado para C++. Embora sua arquitetura de oito camadas (cinco convolucionais e três totalmente conectadas), com o uso da função ReLU e *dropout*, seja idêntica em conceito ao AlexNet, a importância do CaffeNet reside na sua função como o primeiro modelo de referência de alto desempenho amplamente disseminado, acelerando o desenvolvimento de redes neurais convolucionais subsequentes e consolidando o Caffe como um *backend* de fácil implementação no início da era do Deep Learning [JIA et al., 2014].

O algoritmo CaffeNet em efeitos práticos, assemelha-se ao modelo implementado do AlexNet, porém possui uma inversão entre algumas camadas de normalização e *pooling*. Essa modificação da estrutura tem o intuito de verificar se a ordem de aplicação de algumas camadas, afeta o desempenho do modelo como um todo. O algoritmo pode ser visto em sua totalidade no Apêndice C. A Figura 25 mostra a estrutura do algoritmo.

O algoritmo FCTP-Net é um modelo proposto por [WU; ZHU; GUO, 2020] e se assemelha ao modelo CaffeNet, de modo a retirar uma camada de convolução, obtendo uma melhoria de 7% de acurácia . O Apêndice D contem a estrutura detalhada do algoritmo. A Figura 26 mostra o resumo do algoritmo.

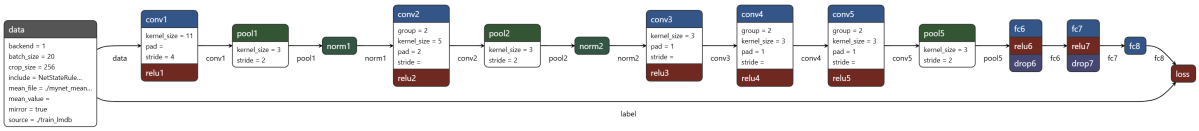


Figura 25 – Modelo de classificação CaffeNet. Fonte: Autor

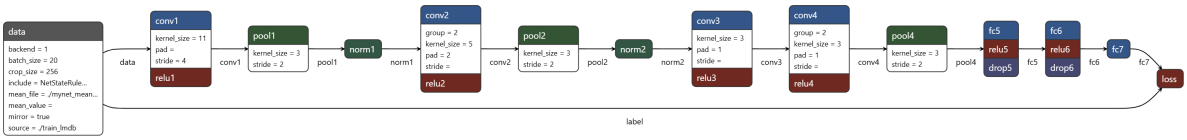


Figura 26 – Modelo de classificação FCTP-Net. Fonte: Autor

5 Treinamento

Para a realização do treinamento dos modelos, as imagens do *dataset* foram normalizadas com tamanho de 280x208 pixels e separadas em lotes (*batch*) de 64 imagens. Do *dataset* de 7215 imagens, 5051 (70%) foram separadas para treinamento e 2164 (30%) para validação.

Os modelos LeNet e AlexNet foram compilados utilizando a otimização com o algoritmo de gradiente adaptativo, ou *Adaptive Gradient Algorithm (AdaGrad)* em inglês. Este algoritmo de otimização utiliza taxas de aprendizado específicas para cada peso, mantendo um histórico dos gradientes passados. Em históricos que possuem um gradiente muito grande para um peso, o *AdaGrad* diminui a taxa de aprendizado para esse peso, reduzindo a chance de ocorrerem variações muito grandes na taxa de aprendizado, tornando o treinamento mais estável. Em históricos de pesos menores, a taxa de aprendizado é menos reduzida, impedindo que o modelo negligencie características que são menos frequentes, mas que são importantes [DUCHI; HAZAN; SINGER, 2011].

Já os modelos CaffeNet e FCTP-Net foram compilados utilizando a otimização com o algoritmo de gradiente descendente estocástico, ou *Stochastic Gradient Descent* (SGD) em inglês. Este algoritmo de otimização permite que o modelo gere uma estimativa do gradiente de modo estocástico, através da média do gradiente calculado em mini-lotes. Por utilizar mini-lotes e ser estocástico, o tempo de processamento a cada atualização de pesos é muito menor, permitindo uma convergência de valores mesmo quando os dados de treinamento são muito grandes. Entretanto, o algoritmo é capaz de convergir rapidamente o modelo para um mínimo local [GOODFELLOW; BENGIO; COURVILLE, 2016].

Os algoritmos de otimização para os modelos foram escolhidos de acordo com os melhores desempenhos obtidos durante as diversas sessões de treinamento realizadas. Os modelos utilizaram a acurácia e a função de perda por entropia cruzada categórica como métricas de avaliação de desempenho, de modo que a evolução da função de perda durante o treinamento nos informa se o modelo está aprendendo de forma eficiente e a acurácia indica a eficácia na tarefa final de classificação.

A acurácia quantifica a relação entre o número de previsões corretas e a quantidade total de previsões do modelo, de acordo com Equação 5.1. Essa métrica resume de forma simplificada o desempenho geral do modelo treinado.

A função de perda (*loss*) por sua vez quantifica o erro do modelo, ou seja, a discrepância entre as previsões do modelo e os valores reais. Durante o treinamento, a CNN faz uma previsão, a função de *loss* avalia o erro dessa previsão e o algoritmo de otimização utiliza esse valor para ajustar os pesos e valores de viés da rede para reduzir o

valor da função de perda. Para os modelos de classificação de multi-classe estudados neste trabalho, utilizou-se a função de perda por entropia cruzada categórica, ou *Categorical Cross Entropy* em inglês, que mede a discrepância entre a distribuição de probabilidade prevista pelo modelo e a distribuição de probabilidade real [NIELSEN, 2015]. Para calcular a função de perda, é necessário que o rótulo real da imagem, chamado de *ground truth*, seja formatado de maneira compatível com a distribuição de probabilidades das classes estimadas pela rede. Para modelos de classificação multiclasse, isso é realizado através da técnica de codificação *One-Hot*. Essa codificação converte o rótulo categórico de uma classe em um vetor binário onde o tamanho do vetor é igual ao número total de classes. Neste vetor, apenas o índice correspondente à classe correta recebe o valor unitário (valor "*hot*"), enquanto todas as demais posições recebem o valor nulo. A Equação 5.2 descreve a expressão dessa função de perda, onde y_i corresponde ao valor real da classe obtido através do *one hot encoding*, e $\hat{y}_i$ dá a probabilidade prevista pelo modelo para a i-ésima classe.

$$\text{Acurácia} = \frac{\text{Previsões corretas}}{\text{Total de previsões}} \tag{5.1}$$

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{5.2}$$

Os modelos foram treinados por 100 épocas, ou *epochs* em inglês. Espera-se que a cada nova época o sistema ajuste os pesos dos neurônios para diminuir o valor da função de perda. Entretanto, a realização do treinamento dos modelos por um número excessivo de épocas pode resultar em modelos com pesos que produzem bom desempenho em termos da *loss function*, apenas para os dados de treinamento. Consequentemente, nesse cenário, o modelo passa a aumentar sua função *loss* para os dados de validação, ou seja, começa a perder sua capacidade de generalizar, isto é, a confiabilidade nas predições realizadas para dados inéditos ao sistema. Ao final do treinamento dos modelos, para evitar a perda de generalização do sistema, fenômeno denominado sobreajuste, ou *overfitting*, são restaurados os modelos que possuem o menor valor de *loss* para os dados de validação. Essa técnica é chamada de parada antecipada, ou *early stopping* em inglês [GOODFELLOW; BENGIO; COURVILLE, 2016].

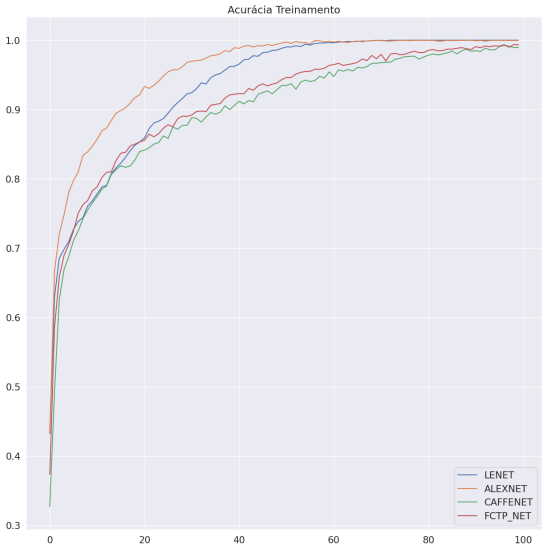
Em seguida, foram geradas as matrizes confusão com mapa de calor, para o treinamento e a validação. A matriz confusão, possui no seu eixo horizontal as classes preditas pelo sistema e no eixo vertical as classes reais da base de dados. Os valores na diagonal principal representam o número de predições em que o modelo acertou. Para uma avaliação multiclasse, esses valores correspondem aos valores Verdadeiros apenas quando a métrica é calculada sob uma perspectiva "um contra todos", onde a classe em questão é a positiva e todas as outras são as negativas [JURAFSKY; MARTIN, 2025].

A matriz confusão com mapa de calor, auxilia a identificar as relações entre quais classes preditas são confundidas com as classes reais. O mapa de calor tem uma cor mais clara em pontos de maior concentração de acertos ou erros, facilitando a visualização dos pontos de maior atenção.

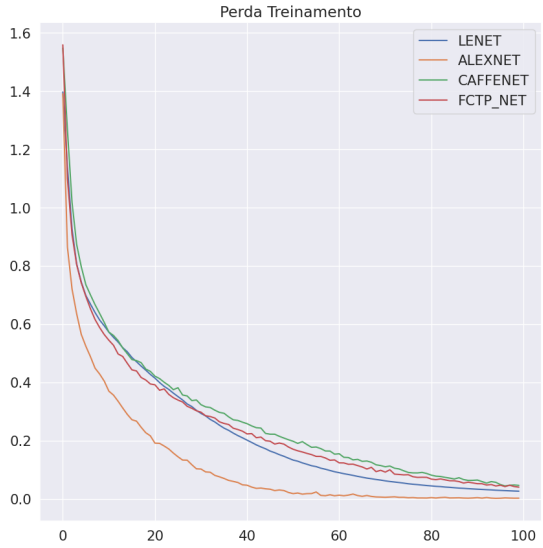
6 Resultados e Discussão

Os algoritmos em linguagem *python* foram implementados no ambiente online Google Colaboratory, pois esta plataforma permite a execução de códigos, sem a necessidade de que o desenvolvedor possua uma máquina potente ou um hardware específico para o treinamento e teste de redes neurais. Conforme descrito no Capítulo 5 de treinamento, foram utilizadas aproximadamente 7200 impressões digitais, em uma proporção de 70% para treinamento e 30% para validação dos modelos.

A Figura 27a e a Figura 27b contêm as curvas de acurácia e *loss* do treinamento. Por sua vez a Figura 28a e a Figura 28b contêm as curvas de acurácia e *loss* da validação. A Tabela 1 compila os valores de acurácia de treinamento e acurácia de validação, de acordo com os melhores valores de perda de validação obtidos durante o treinamento, através da técnica de *early stopping*.



(a) Acurácia de treinamento.



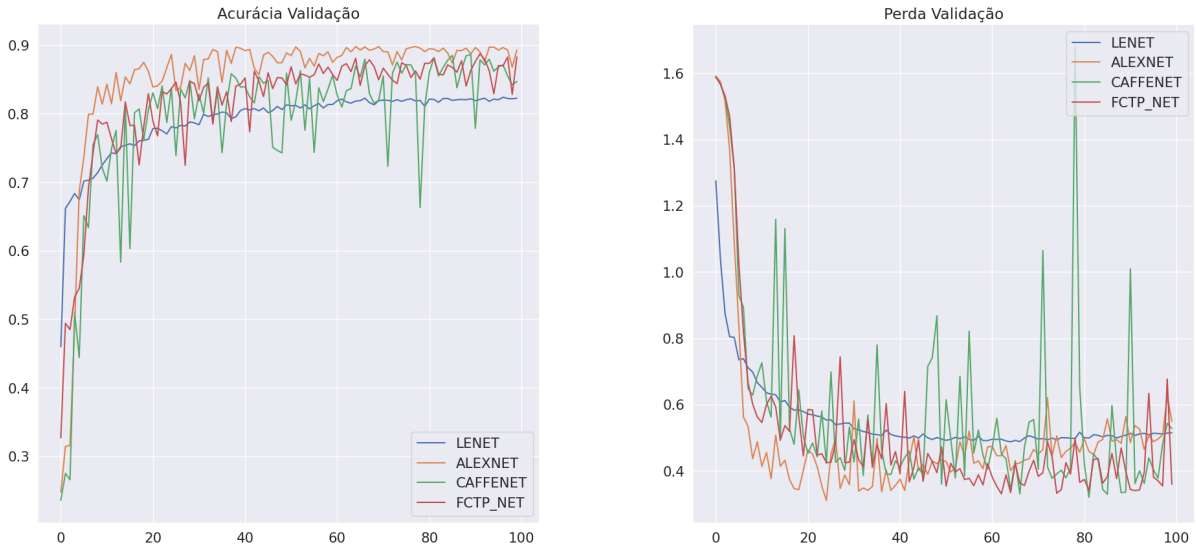
(b) Loss de treinamento.

Figura 27 – Curvas de acurácia e *loss* de treinamento dos modelos. Fonte: Autor

Tabela 1 – Resultados do Treinamento de Modelos de Redes Neurais

Modelo	Melhor época	Acurácia treino	Acurácia validação
LeNet	65	0,9980	0,8184
AlexNet	25	0,9502	0,8863
CaffeNet	82	0,9856	0,8817
FCTP-NET	63	0,9628	0,8729

Baseando-se nos valores obtidos durante a validação dos modelos, os algoritmos LeNet, AlexNet e CaffeNet, obtiveram performances acima do esperado quando comparado



(a) Acurácia de validação.

(b) Loss de Validação.

Figura 28 – Curvas de acurácia e *loss* de validação dos modelos. Fonte: Autor

aos valores apresentados na Tabela 2, obtidos por [WU; ZHU; GUO, 2020]. Já o algoritmo FCTP-Net obteve um valor semelhante.

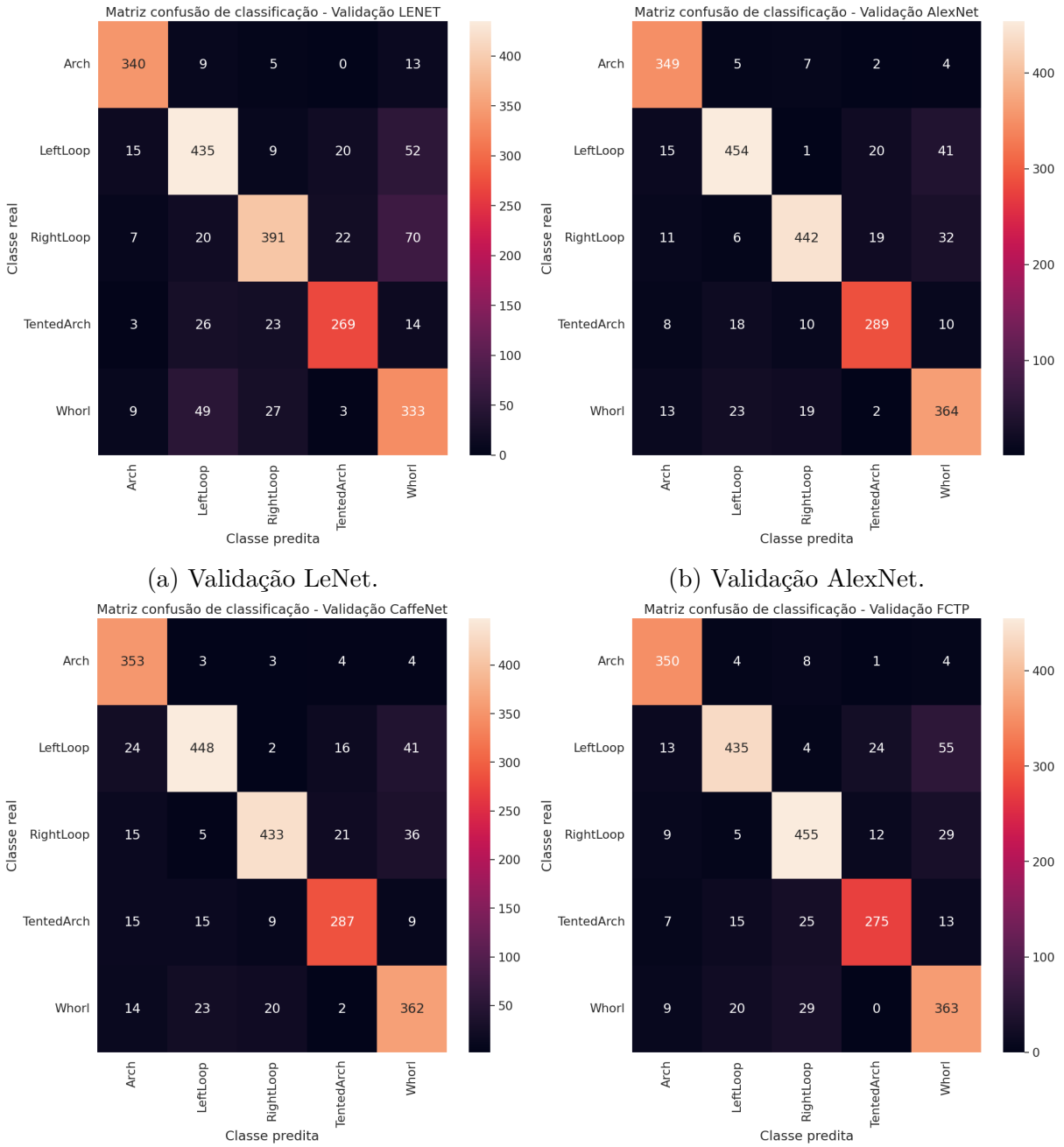
Tabela 2 – Resultados de treinamento obtidos em [WU; ZHU; GUO, 2020]

Modelo	Acurácia treino	*Loss* treino
LeNet	0,1664	1,8745
AlexNet	0,4703	1,2516
CaffeNet	0,8263	0,5018
FCTP-NET	0,9314	0,2886

As variações de performance dos algoritmos em relação à [WU; ZHU; GUO, 2020] ocorreram possivelmente devido à diferença nos *datasets* de treinamento, onde foram coletadas 100000 impressões digitais, das quais 21300 foram selecionadas para treinamento e 1200 para testes, aproximadamente quatro vezes a quantidade utilizada neste trabalho, possivelmente gerando modelos com menor *overfitting* e uma generalização superior.

A Figura 29 mostra as matrizes confusão de validação dos modelos. Observa-se que, após o treinamento, todos os modelos apresentaram uma distribuição elevada de acertos na diagonal principal, com valores de acurácia apresentados na Tabela 1. Verifica-se também que os modelos realizaram uma quantidade maior de predições incorretas na classe *whorl*, sendo que o esperado seria uma confusão maior entre as classes *arch* e *tented arch*, por suas características fisiológicas muito semelhantes.

Após o término do treinamento, os algoritmos podem então passar a prever com certa confiabilidade os tipos de biometrias em novas imagens. Possibilitando, como exemplificado na Figura 30, pesquisar imagens em plataformas como o Google, em seguida, um



(a) Validação LeNet.

(b) Validação AlexNet.

(c) Validação CaffeNet.

(d) Validação FCTP-Net.

Figura 29 – Matriz confusão de validação dos modelos estudados. Fonte: Autor

trecho de código utiliza os modelos treinados, para predizer a porcentagem de confiança da imagem pertencer a cada classe. A Figura 31 ilustra a execução deste código, possibilitando observar que, apenas os modelos AlexNet, CaffeNet e FCTP-Net acertaram o tipo de biometria como *whorl*. O modelo LeNet, classificou erroneamente a biometria como *right loop*, obtendo probabilidades distribuídas de modo aproximadamente uniforme entre quatro classes. Isso corrobora a observação de que o modelo possui menor acurácia e mais erros distribuídos em sua matriz confusão.

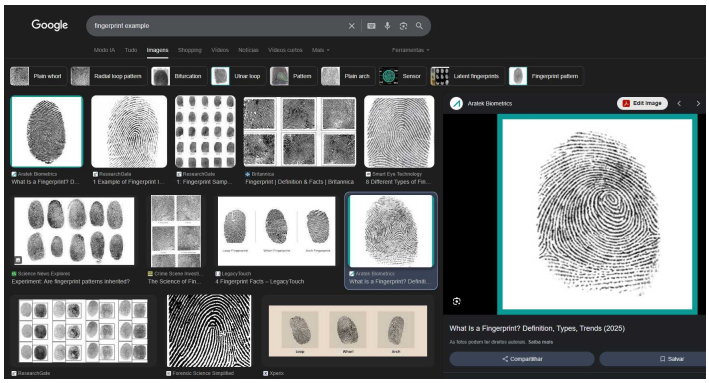


Figura 30 – Exemplo de pesquisa de biometria do tipo *Whorl* no Google. Fonte: Autor

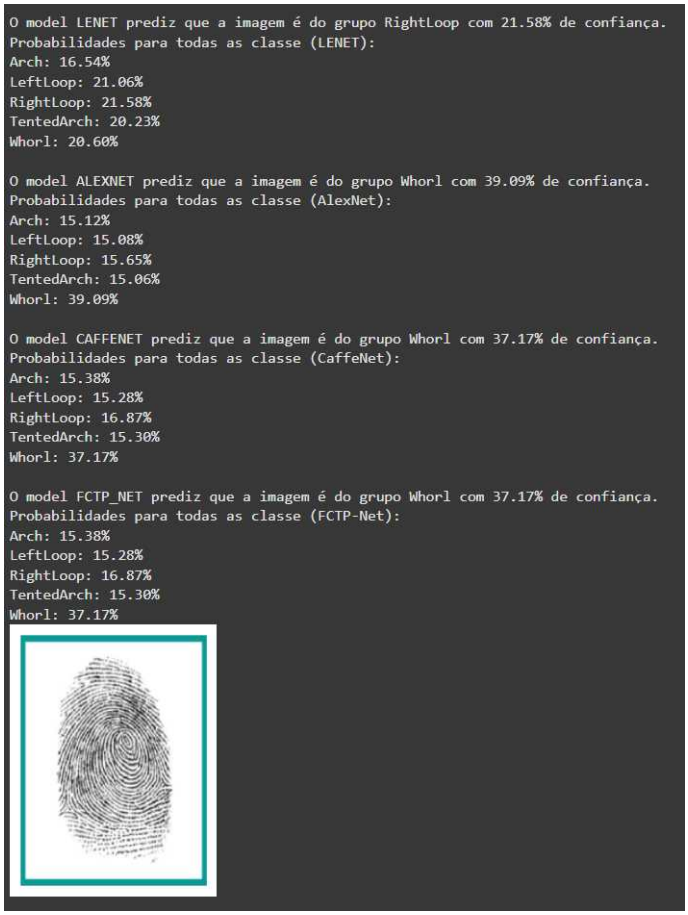


Figura 31 – Exemplo de predição dos modelos para uma biometria aleatória do tipo *Whorl*. Fonte: Autor

Com as predições obtidas através dos modelos de classificação, são realizados testes, de acordo com o explicado na Seção 2.3, para verificar o tempo de busca que uma biometria aleatória levaria para ser verificada no banco de dados. Em um caso hipotético, buscando avaliar apenas a velocidade de busca, sem utilizar impressões digitais e assumindo acurácia de 100% , foi criado um banco de dados com 1 milhão de registros unicos, cada uma com uma identificação própria(ID). Este banco de dados, chamado de banco 1, não possui ordenação por tipos de biometrias, apenas há ordenação dos IDs sequencialmente, por exemplo, a impressão digital de ID 1000 estará na posição 1000 do índice da base de dados. Este banco de dados, então é copiado duas vezes, de modo que um dos bancos, chamado de banco 2, possuirá a ordenação das impressões digitais e busca de acordo com a distribuição de 33,8%, 32,7%, 27,9%, 3,7% e 2,9%, respectivamente para as classes *Left Loop*, *Right Loop*, *Whorl*, *Arch* e *Tented Arch*. Neste caso uma impressão digital de exemplo, com ID 1000 e do classe *arch*, possivelmente estará em um índice superior à 940000. O outro banco, chamado de banco 3 ou banco classificado, possui a mesma distribuição de biometrias do banco sequencial, porém a ordem de busca será dada pelo algoritmo de classificação.

Baseando-se no tempo de execução de equipamentos comerciais [Suprema Inc., 2018], que realizam cerca de 5 mil comparações por segundo, o tempo de busca e comparação de cada biometria foi fixado em 200 microssegundos.

Entende-se que, quanto maior o banco de dados, pior será o desempenho da busca no banco de dados 1, quando a impressão digital buscada estiver em um índice próximo ao final da base de dados. O mesmo pode ser aplicado para a busca no banco 2, que sempre priorizará as classes com maior distribuição de usuários. Por exemplo, gerando buscas desnecessárias em pelo menos um terço do banco de dados quando a biometria não for da classe *LL*. Logo, espera-se que a busca classificada no banco 3 possuirá as melhores médias de tempo de busca.

Para testar as hipóteses acima, foram realizadas aproximadamente 10 mil execuções de busca e comparação de registros aleatórios. De modo que, cada busca gerava registro, uma impressão digital, com ID e classe randômicos. Na busca realizada no banco 1, o algoritmo utiliza apenas o ID do registro e o compara com o ID cadastrado no índice da base de dados. Na busca realizada no banco 2, o algoritmo utiliza apenas o ID do registro e realiza a comparação com os IDs cadastrados, seguindo a ordem de distribuição de classes do maior para o menor, ou seja, inicia a busca na classe *left loop* e se necessário termina na classe *tented arch*. Na busca realizada no banco 3, o algoritmo prediz a classe do registro e compara o ID do registro com o ID cadastrado na base de dados, seguindo a ordem de predição das classes, de maior probabilidade para o menor probabilidade. Após a realização das buscas mencionadas, o algoritmo calcula a média de tempo gasto em cada banco de dados.

A Figura 32 mostra a distribuição de tempos de busca para os três bancos de

dados para 10000 iterações, obtendo valores médios de 97898 milissegundos (98 segundos) para o banco 1, 99044 milissegundos (99 segundos) para o banco 2 e 35557 milissegundos (35 segundos) para a busca no banco 3 com classificação, com desvio padrão de 58815 milissegundos, 58564 milissegundos e 33722 milissegundos, respectivamente.

O tempo médio de busca no Banco 1 é relativamente alto, mas esperado, considerando que algoritmo realiza a comparação sequencialmente nos índices da base de dados, de modo que, o total de registros (1 milhão) impacta consideravelmente o tempo médio. No Banco 2, onde a busca é sequencial dentro de cada classe, obteve-se um tempo médio muito parecido com o do Banco 1. Isso acontece porque, apesar de ser sequencial, o número de registros por classe não deve variar tanto a ponto de causar grandes diferenças no desempenho em comparação com o Banco 1. O Banco 3 tem um tempo médio significativamente mais baixo. Isso sugere que a ordem das classes definida pelo modelo de classificação teve um impacto positivo na busca, acelerando a localização do registro. A classificação organizou as classes de forma que a busca fosse mais eficiente, aproveitando a distribuição interna para reduzir o tempo de busca. Com a acurácia de aproximadamente 89% dos modelos, uma redução de aproximadamente 65,5% em relação às busca no banco 1 não ordenado e no banco 2 com ordenação.

O desvio padrão é relativamente alto, significando que houve grandes variações nos tempos de busca para os bancos 1 e 2, isso se deve a aleatoriedade dos registros, o que pode ser verificado pela distribuição uniforme dos tempos ao longo do eixo vertical da Figura 32. O desvio padrão no banco 3, foi bem mais baixo, indicando que a classificação contribuiu para reduzir a variabilidade dos tempos de busca.

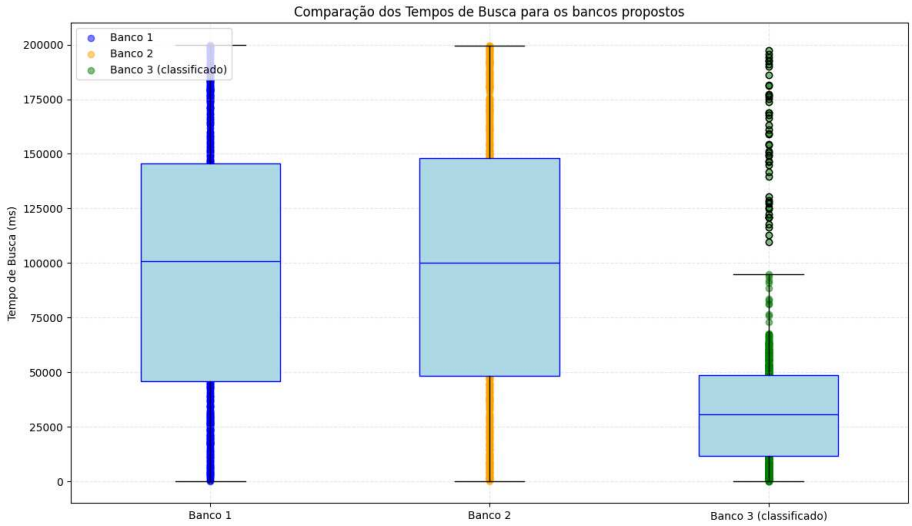


Figura 32 – Distribuição de tempo de busca para 10 mil buscas nos bancos de dados propostos. Fonte: Autor

7 Considerações Finais e Trabalhos Futuros

7.1 Considerações Finais

Este projeto buscou dois objetivos principais, o primeiro de testar algoritmos de classificação de biometrias baseados em redes neurais, o segundo de utilizar esses algoritmos para melhorar o desempenho de busca e comparação de biometrias em bancos de dados. Os algoritmos de classificação resultaram em modelos de fácil utilização (como carregar a imagem da impressão digital) e capazes de predizer com eficácia a classe de impressões digitais. A utilização desses modelos para acelerar a busca de uma biometria em um banco de dados apresentou médias de tempo menores do que outros métodos de busca. Logo, pode-se concluir que a utilização de tais algoritmos reduz o tempo necessário de processamento e ocupação do sistema, possibilitando reduzir custos operacionais, que em larga escala (milhares de comparações de biometrias por dia), podem ser relevantes para uma empresa.

Os testes e simulações dos algoritmos estudados apresentaram resultados positivos, com valores de acurácia de treinamento superiores a 91% e acurácia de validação superiores à 80%, desempenho semelhante ao proposto por [WU; ZHU; GUO, 2020]. Obteve-se, em simulações, uma redução de aproximadamente 65% no tempo de busca de impressões digitais realizadas no banco de dados com o emprego dos modelos de classificação.

7.2 Trabalhos Futuros

Para dar continuidade ao projeto, sugere-se o estudo de metodologias alternativas de classificação de impressões digitais atualmente utilizadas.

É também possível a integração de um algoritmo de comparação de biometria ao sistema de classificação. Essa otimização visa reduzir ainda mais a latência de resposta percebida pelo usuário final.

Após o treinamento dos modelos, será viável implementar e testar o sistema em aplicações embarcadas (como microcontroladores com ou sem NPU) utilizando o TensorFlow Lite. Isso permitirá o desenvolvimento de modelos específicos para que a inferência seja realizada diretamente nos dispositivos.

Referências

ABADI, M. et al. TensorFlow: a system for Large-Scale machine learning. In: *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. [S.l.: s.n.], 2016. p. 265–283. Citado na página 26.

AGGARWAL, C. C. et al. *Neural networks and deep learning*. [S.l.]: Springer, 2018. v. 10. Citado na página 12.

ANDRADE, E. T. d. *Treinamento de Redes Neurais Convolucionais com Uso de Imagens Sintéticas e Técnicas de Interpretabilidade*. Dissertação (Dissertação de Mestrado) — Universidade Federal do ABC, Santo André, 2022. Citado 3 vezes nas páginas 4, 17 e 18.

BISONG, E. et al. *Building machine learning and deep learning models on Google cloud platform*. [S.l.]: Springer, 2019. Citado na página 26.

BOOK, D. *Data Science Academy. Deep Learning Book*. 2023. Citado 2 vezes nas páginas 4 e 12.

CHOLLET, F. The limitations of deep learning. *Deep learning with Python*, 2017. Citado na página 26.

CORMEN, T. et al. *Book: introduction to algorithms*. [S.l.]: MIT Press, 2009. Citado na página 6.

DAUGMAN, J. New methods in iris recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 37, n. 5, p. 1167–1175, 2007. Citado 2 vezes nas páginas 4 e 2.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, v. 12, n. 7, 2011. Citado na página 29.

FERREIRA, M. F. d. C. et al. Avaliaçao do uso redes neurais convolucionais para identificaçao de lesoes cariosas dentárias. In: SBC. *Simpósio Brasileiro de Computação Aplicada à Saúde (SBCAS)*. [S.l.], 2023. p. 473–478. Citado 2 vezes nas páginas 4 e 14.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 9 vezes nas páginas 4, 13, 15, 16, 17, 19, 20, 29 e 30.

HASSAN, H.; KIM, H.-W. Cmos capacitive fingerprint sensor based on differential sensing circuit with noise cancellation. *Sensors*, MDPI, v. 18, n. 7, p. 2200, 2018. Citado 2 vezes nas páginas 4 e 9.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. *International conference on machine learning*. [S.l.], 2015. p. 448–456. Citado 2 vezes nas páginas 18 e 19.

JAIN, A. K.; ROSS, A.; PRABHAKAR, S. An introduction to biometric recognition. *IEEE Transactions on circuits and systems for video technology*, IEEE, v. 14, n. 1, p. 4–20, 2004. Citado na página 2.

JIA, Y. et al. Caffe: Convolutional architecture for fast feature embedding. In: *Proceedings of the 22nd ACM international conference on Multimedia*. [S.l.: s.n.], 2014. p. 675–678. Citado na página 27.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd. ed. [s.n.], 2025. Online manuscript released August 24, 2025. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>. Citado na página 30.

KANDEL, E. R. *Principles of neural science*. [S.l.]: McGraw-hill, 2000. Citado na página 11.

KATEGARU, S. *Convolutional Neural Network | Deep Learning*. 2020. Disponível em: <https://developersbreach.com/convolution-neural-network-deep-learning/>. Citado 3 vezes nas páginas 4, 16 e 18.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, v. 25, 2012. Citado 2 vezes nas páginas 19 e 27.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 2002. Citado na página 26.

LEE, H. et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th annual international conference on machine learning*. [S.l.: s.n.], 2009. p. 609–616. Citado 3 vezes nas páginas 4, 13 e 14.

MALTONI, D. Generation of synthetic fingerprint image databases. *Automatic fingerprint recognition systems*, Springer, p. 361–384, 2004. Citado 2 vezes nas páginas 5 e 21.

MALTONI, D. et al. *Handbook of fingerprint recognition*. [S.l.]: Springer, 2009. v. 2. Citado 7 vezes nas páginas 4, 1, 2, 3, 8, 9 e 20.

MILLAR, J. et al. Benchmarking ultra-low-power μnpus. In: *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*. [S.l.: s.n.], 2025. p. 1060–1074. Citado na página 4.

NARANJO, J. L. *Inteligência computacional aplicada na geração de respostas impulsivas bi-auriculares e em aurilização de salas*. Tese (Doutorado) — Universidade do Estado do Rio de Janeiro, 05 2014. Citado 2 vezes nas páginas 4 e 12.

NIELSEN, M. A. *Neural networks and deep learning*. [S.l.]: Determination press San Francisco, CA, USA, 2015. v. 25. Citado na página 30.

RIM, B.; KIM, J.; HONG, M. Fingerprint classification using deep learning approach. *Multimedia Tools and Applications*, Springer, v. 80, n. 28, p. 35809–35825, 2021. Citado 2 vezes nas páginas 3 e 8.

Suprema Inc. *SFM Slim: Ultra-Slim FAP20 Fingerprint Module*. 16F Parkview Tower, 248, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13554, Rep. of KOREA: [s.n.], 2018. Product brochure. Document number: DHL-SFMSlim-180502-02-EN. Disponível em: <http://www.supremainc.com>. Citado na página 36.

TANG, H.-Y. et al. 3-d ultrasonic fingerprint sensor-on-a-chip. *IEEE Journal of Solid-State Circuits*, IEEE, v. 51, n. 11, p. 2522–2533, 2016. Citado 2 vezes nas páginas 4 e 10.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. *Proceedings of the xxix conference on graphics, patterns and images*. [S.l.], 2016. v. 1, n. 4. Citado 4 vezes nas páginas 13, 15, 16 e 18.

WU, F.; ZHU, J.; GUO, X. Fingerprint pattern identification and classification approach based on convolutional neural networks. *Neural Computing and Applications*, Springer, v. 32, n. 10, p. 5725–5734, 2020. Citado 4 vezes nas páginas 6, 27, 33 e 38.

YANG, S.; SAKIYAMA, K.; VERBAUWHEDE, I. Efficient and secure fingerprint verification for embedded devices. *EURASIP Journal on Advances in Signal Processing*, Springer, v. 2006, n. 1, p. 058263, 2006. Citado na página 4.

ZHOU, Y.; LIANG, Y.; TAN, P. Design of an intelligent laboratory facial recognition system based on expression keypoint extraction. *IEEE Access*, p. 1–1, 2023. Citado 3 vezes nas páginas 4, 1 e 2.

Apêndices

APÊNDICE A – Arquitetura LeNet

- **Camada de Convolução:** 20 filtros com *kernel* de 5x5 e *stride* de 1x1.
- **Camada de MaxPooling:** *kernel* de 2x2 e *stride* de 2x2.
- **Camada de Convolução:** 50 filtros com *kernel* de 5x5 e *stride* de 1x1.
- **Camada de MaxPooling:** *kernel* de 2x2 e *stride* de 2x2.
- **Camada Fully Connected:** 500 filtros com função de ativação ReLU.
- **Camada Fully Connected:** Com função de ativação SoftMax, que fornece a probabilidade das 5 classes preditas.

APÊNDICE B – Arquitetura AlexNet

- **Camada de Convolução:** 96 filtros com *kernel* de 11x11 e *stride* de 4x4.
- **Camada de BatchNormalization:** normalização de ativações mais relevantes.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada de Convolução:** 256 filtros com *kernel* de 5x5, *stride* de 1x1 e função de ativação Relu.
- **Camada de BatchNormalization:** normalização de ativações mais relevantes.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada de Convolução:** 384 filtros com *kernel* de 3x3, *stride* de 1x1 e função de ativação Relu.
- **Camada de Convolução:** 384 filtros com *kernel* de 3x3, *stride* de 1x1 e função de ativação Relu.
- **Camada de Convolução:** 256 filtros com *kernel* de 3x3, *stride* de 1x1 e função de ativação Relu.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada Fully Connected:** 4096 filtros com função de ativação ReLU.
- **Camada de Dropout:** desativação de 50% de neurônios aleatórios da camada anterio.
- **Camada Fully Connected:** 4096 filtros com função de ativação ReLU.
- **Camada de Dropout:** desativação de 50% de neurônios aleatórios da camada anterior.
- **Camada Fully Connected:** Com função de ativação SoftMax, que fornece a probabilidade das 5 classes preditas.

APÊNDICE C – Arquitetura CaffeNet

- **Camada de Convolução:** 96 filtros com *kernel* de 11x11 e *stride* de 4x4.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada de BatchNormalization:** normalização de ativações mais relevantes.
- **Camada de Convolução:** 256 filtros com *kernel* de 5x5, *stride* de 1x1 e função de ativação Relu.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada de BatchNormalization:** normalização de ativações mais relevantes.
- **Camada de Convolução:** 384 filtros com *kernel* de 3x3, *stride* de 1x1 e função de ativação Relu.
- **Camada de Convolução:** 384 filtros com *kernel* de 3x3, *stride* de 1x1 e função de ativação Relu.
- **Camada de Convolução:** 256 filtros com *kernel* de 3x3, *stride* de 1x1 e função de ativação Relu.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada Fully Connected:** 4096 filtros com função de ativação ReLU.
- **Camada de Dropout:** desativação de 50% de neurônios aleatórios da camada anterio.
- **Camada Fully Connected:** 4096 filtros com função de ativação ReLU.
- **Camada de Dropout:** desativação de 50% de neurônios aleatórios da camada anterior.
- **Camada Fully Connected:** Com função de ativação SoftMax, que fornece a probabilidade das 5 classes preditas.

APÊNDICE D – Arquitetura FCTP-Net

- **Camada de Convolução:** 96 filtros com *kernel* de 11x11 e *stride* de 4x4.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada de BatchNormalization:** normalização de ativações mais relevantes.
- **Camada de Convolução:** 256 filtros com *kernel* de 5x5, *stride* de 1x1 e função de ativação Relu.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada de BatchNormalization:** normalização de ativações mais relevantes.
- **Camada de Convolução:** 384 filtros com *kernel* de 3x3, *stride* de 1x1 e função de ativação Relu.
- **Camada de Convolução:** 384 filtros com *kernel* de 3x3, *stride* de 1x1 e função de ativação Relu.
- **Camada de MaxPooling:** *kernel* de 3x3 e *stride* de 2x2.
- **Camada Fully Connected:** 4096 filtros com função de ativação ReLU.
- **Camada de Dropout:** desativação de 50% de neurônios aleatórios da camada anterio.
- **Camada Fully Connected:** 4096 filtros com função de ativação ReLU.
- **Camada de Dropout:** desativação de 50% de neurônios aleatórios da camada anterior.
- **Camada Fully Connected:** Com função de ativação SoftMax, que fornece a probabilidade das 5 classes preditas.