UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Lahiri Jumonji Godinho

# Avaliação do desempenho de Redes Neurais Convolucionais para o reconhecimento biométrico da região periocular utilizando *Transfer Learning*

São Carlos

2019

**Lahiri Jumonji Godinho**

# Avaliação do desempenho de Redes Neurais Convolucionais para o reconhecimento biométrico da região periocular utilizando *Transfer Learning*

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Sistemas de Energia e Automação, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Adilson Gonzaga

**São Carlos**

**2019**

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

GG585a

Godinho, Lahiri Jumonji
Avaliação do desempenho de Redes Neurais Convolucionais para o reconhecimento biométrico da região periocular utilizando Transfer Learning / Lahiri Jumonji Godinho; orientador Adilson Gonzaga. São Carlos, 2019.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019.

1. Reconhecimento da região periocular. 2. CNN. 3. Transfer Learning. 4. Fine Tunning. 5. Íris a distância. 6. CASIA-IrisV4. 7. Aprendizagem profunda. 8. Aprendizagem de máquina. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

# FOLHA DE APROVAÇÃO

**Nome: Lahiri Jumonji Godinho**

**Título: "Avaliação do desempenho de Redes Neurais Convolucionais para o reconhecimento biométrico da região periocular utilizando Transfer Learning"**

**Trabalho de Conclusão de Curso defendido e aprovado em 25/11/2019,**

**com NOTA 8,0 ( OITO, ZERO ), pela Comissão Julgadora:**

***Prof. Associado Adilson Gonzaga - Orientador - SEL/EESC/USP***

***Dra. Carolina Toledo Ferraz - Pós-Doutorado/Faculdade Campo Limpo Paulista***

***Mestre Osmando Pereira Junior - Doutorando - SEL/EESC/USP***

**Coordenador da CoC-Engenharia Elétrica - EESC/USP:**
**Prof. Associado Rogério Andrade Flauzino**

*Aos meus pais, Joaquim e Tomigracy, pela fé, compreensão, carinho e apoio incondicional.*

## AGRADECIMENTOS

Primeiramente, aos meus pais Joaquim e Tomigracy, que sempre acreditaram em mim. Sem seus conselhos, inspiração e apoio incondicional nada disso seria possível.

A Virgínia, que sempre esteve lá quando eu mais precisei e que nunca desistiu de mim. Pela paciência, carinho e suporte emocional.

A Ana Carolina, por me ajudar continuamente na revisão desta monografia e pelo ritmo e motivação proporcionados.

Ao meu orientador Adilson, pela paciência e compreensão nessa etapa final do curso.

A Carolina e Nicole, pela continua assistência com edição de imagens.

Ao Luis e a Sofia, pela companhia nas horas de necessidade e divertimento nas horas de descontração.

A toda a minha família e amigos, que me proporcionaram alegria, companhia e puderam acompanhar a minha jornada, torcer e comemorar cada conquista comigo.

A todos os professores que participaram da minha formação educacional desde o ingresso na escola, e contribuíram com meu aprendizado para que eu pudesse chegar até aqui.

*"Aqueles que conseguem imaginar qualquer coisa*
*conseguem criar o impossível."*
*Alan Turing*

# RESUMO

GODINHO, L. J. **Avaliação do desempenho de Redes Neurais Convolucionais para o reconhecimento biométrico da região periocular utilizando *Transfer Learning***. 2019. 87p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

O reconhecimento biométrico tem diversas aplicações atualmente por ser uma maneira segura de identificação. Dentre as diversas maneiras de se fazer o reconhecimento biométrico, o método que vem se destacando mais é a utilização de redes neurais convolucionais (CNN). Apesar de apresentarem bons resultados em diversas aplicações, a etapa de treinamento requer um conjunto de dados composto por muitas amostras e exige em longo tempo de execução. Estes pré-requisitos do treinamento se tornam um problema pois nem sempre existe um conjunto de dados com muitas amostras para a aplicação específica. Neste trabalho, é avaliada a técnica de *transfer learning* onde uma rede pré treinada é utilizada de base para uma rede nova adaptada ao problema. Com esta técnica, é possível treinar a nova rede rapidamente e utilizando uma quantidade menor de dados. Foram escolhidas quatro redes pré treinadas, baseadas em seu desempenho no *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC). Estas redes foram sintonizadas por *transfer learning* com um subconjunto de imagens da base CASIA-IrisV4, coletada pela *Chinese Academy of Sciences' Institute of Automation* (CASIA).As CNNs testadas apresentaram acurácias elevadas, de 91% a 95%, quando as duas regiões perioculares (olho esquerdo e direito) são utillizadas no treinamento. Além disso, a inversão por software das imagens das regiões perioculares direita ou esquerda na fase de teste reduziu o desempenho das redes para acurácias abaixo de 50% quando um olho invertido é testado em uma rede treinada apenas com o olho oposto. O desempenho das CNNs, no entando, apresenta acurácias próximas de 80% quando o olho invertido é testado em uma rede treinada com a região periocular do mesmo lado da face. Os resultados mostram que as características de uma região periocular (direita ou esquerda) possuem propriedades discriminativas próprias, que são identificadas pelas redes, mesmo em caos de inversão da imagem.

**Palavras-chave**: Aprendizagem de máquina, Aprendizagem Profunda. *Transfer learning. Fine tunning.* CNN. Iris a distância. CASIA-IrisV4. Reconhecimento da região periocular.

## ABSTRACT

GODINHO, L. J. **Performance evaluation of Convolutional Neural Networks for biometric recognition of the periocular region using Transfer Learning**. 2019. 87p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Biometric recognition has a lot of aplications nowadays because it's a safe form of identification. Between the many ways of doing biometric recognition, the method that has stood out the most is the use of convolutional neural networks (CNN). Despite presenting good results in various applications, The training step requires a dataset consisting of many samples and a long execution time. These requirements can become a problem because there isn't always a dataset with many samples for the specific application. In this project, we study the transfer learning technique where a pre-trained network is used as a base to the new network, adapted to the problem. With this technique, it's possible to quickly train the new network using a smaller amount of data. Four pre-trained network were used, based on their performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). These networks were tuned by transfer learning with a subset of images from the database CASIA-IrisV4, collected by the Chinese Academy of Sciences' Institute of Automation (CASIA). The tested CNNs presented high accuracy, from 91% to 95%, when the two periocular regions (left and right eyes) are used in the training. In addition, image inversion by software of right or left periocular regions in the test phase reduced net performance to accuracy below 50% when an inverted eye is tested in a network trained with the opposite eye only. However, the performance of CNNs is close to 80% when the inverted eye is tested in a trained network with the periocular region on the same side of the face. The results show that the characteristics of a periocular region (right or left) have their own discriminative properties that are identified by the network, even in cases of imagem inversion.

**Keywords**: Machine learning. Deep learning. Transfer learning. Fine tunning. CNN. Iris at distance. CASIA-IrisV4. Recognition of the periocular region.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| BOR | Biometric Ocular Recognition |
| CASIA | Chinese Academy of Sciences' Institute of Automation |
| CNN | Convolutional Neural Network |
| FCL | Fully Conected Layer |
| HIR | Heterogeneous iris recognition |
| IAAD | Iris at a distance |
| ILSVRC | ImageNet Large Scale Visual Recognition Challenge |
| IOM | Iris On the Move |
| LRR | Long-Range-Recognition |
| MBGC | Multiple Biometric Grand Challenge |
| NIR | Near Infra Red |
| NIST | National Institute of Standards and Technology |

SUMÁRIO

# 1 INTRODUÇÃO

O objetivo da biometria é a identificação do indivíduo utilizando apenas características físicas ou comportamentais específicas dele, trazendo um grau de confiabilidade maior do que uma senha digitada que pode ser esquecida ou roubada. A face humana é uma peculiaridade biométrica de fácil aquisição e que pode ser usada em diversas aplicações devido à grande quantidade de informação que dela pode ser extraída. Diversas formas de reconhecimento facial já estão sendo utilizadas em sistemas de segurança ao redor do mundo (DESHPANDE, 2016), seja de forma complementar ou substituindo completamente outras formas de identificação (DELAC; GRGIC, 2004).

Em anos mais recentes, vem crescendo o uso de redes neurais na área de visão computacional pelo seu potencial discriminativo em problemas de classificação de imagem (SAEZ-TRIGUEROS; MENG; HARTNETT, 2018). Como reconhecimento facial é um caso específico de classificação de imagens, cresce a cada dia o número de sistemas que estão implementando redes neurais em seus algoritmos de reconhecimento facial.

A face humana apresenta diversas regiões com características discriminativas diferentes, tais como íris, região periocular, textura da pele, formato da boca, do nariz ou das orelhas, cor dos cabelos, etc. Deste modo, uma tendência na biometria facial é utilizar apenas uma dessas características, não somente pela facilidade de aquisição, mas também pela facilidade de disfarce se usada a face como um todo. Baseando-se na capacidade humana de reconhecer outros indivíduos olhando apenas para seus olhos, estuda-se a utilização da região periocular para se fazer o reconhecimento biométrico.

A região periocular trás então algumas das vantagens do reconhecimento pela íris sem a necessidade de que a imagem tenha alta uma resolução. Assim como imagens do rosto inteiro, a região periocular pode ser capturada e reconhecida a longas distancias em ambientes não controlados e sem a cooperação do individuo identificado. Desta forma, a região periocular apresenta vantagens tanto da iris quanto da região da face inteira, trazendo uma opção equilibrada entre ambas.

## 1.1 Motivação

Os métodos tradicionais de reconhecimento periocular se baseiam em algoritmos de pré-processamento, segmentação e extração de características. No entanto, o trabalho de pré-processamento da imagem e comparação das características extraídas com o banco de dados tornam estes algoritmos lentos e dificultam seu uso.

Algoritmos de aprendizado profundo, em contraponto, conseguem aprender sozinhos as características a serem analisadas nas imagens. Estes ainda são capazes de fazer a

identificação sem a necessidade de se consultar um banco de dados, tornando-os mais rápidos, uma vez implementados. Assim, técnicas de aprendizado profundo vem sendo cada vez mais utilizadas em reconhecimento biométrico.

A acurácia dessas técnicas depende da forma que a rede foi treinada e da quantidade de imagens fornecidas durante esta etapa de treinamento. A quantidade ideal de imagens para o treinamento não é exata pois varia com as especificidades de cada problema, entretanto, observando-se arquiteturas de redes previamente desenvolvidas, conclui-se que algumas centenas de milhares de imagens são necessárias para um bom desempenho (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) (TSANG, 2018). O tempo de treinamento também apresenta um desafio, uma vez que outros métodos de reconhecimento biométrico são mais rápidos de se implementar do que redes neurais convolucionais (SAEZ-TRIGUEROS; MENG; HARTNETT, 2018).

Para contornar estes problemas, é possível utilizar uma rede neural convolucional já treinada e apenas sintonizá-la para adaptá-la a um problema específico, permitindo reduzir o número de imagens utilizadas no treinamento da rede, assim como o tempo de processamento. Essa técnica de adaptação de uma rede de um problema para outro chama-se *transfer learning* (LI; HOIEM, 2016) e é utilizada neste trabalho.

## 1.2 Objetivos

O objetivo geral deste trabalho é avaliar redes neurais convolucionais pré treinadas e analisar seus desempenhos quando submetidas a problemas relacionados ao reconhecimento biométrico da região periocular da face humana. As redes escolhidas foram sintonizadas com a técnica de *transfer learning* e submetidas a diversos ensaios usando uma pequena base de imagens de faces humanas adquiridas à distância (BIT, 2010). A região periocular destas imagens faciais foram recortadas e utilizadas como entrada de uma CNN pré-treinada.

Os objetivos específicos deste trabalho são: avaliar a eficiência da técnica de *transfer learning* na redução do conjunto de imagens necessário para treinar uma rede, diminuindo assim seu tempo de processamento; verificar se a região periocular possui características discriminativas suficientes para ser utilizada na biometria; investigar se regiões perioculares de lados diferentes da face humana podem ser utilizadas na fase de teste das CNNs invertendo-se as imagens das mesmas por software (via *flip*) tendo sido a rede treinada apenas com imagens não invertidas.

## 1.3 Organização do trabalho

Este trabalho está organizado em 6 capítulos e 5 apêndices. O primeiro capítulo é a introdução, onde o tema é apresentado e os problemas abordados pela primeira vez. O segundo capítulo trata da região periocular e suas peculiaridades biométricas. O terceiro

capítulo detalha conceitos de aprendizado de máquina e na arquitetura das redes neurais bem como o estado da arte e trabalhos relacionados ao tema. No quarto capítulo são apresentados os materiais e métodos utilizados neste trabalho, os estudos que foram planejados e a motivação por trás deles. No quinto capítulo os resultados dos ensaios são apresentados e discutidos. No sexto capítulo conclui-se o trabalho com as considerações finais e sugestões para trabalhos futuros relacionados ao tema. Por fim, nos apêndices, são apresentados os códigos utilizados durante o trabalho.

# 2 BIOMETRIA DA REGIÃO PERIOCULAR

Os maiores problemas dos sistemas biométricos estão relacionados à sua aceitação devido a equipamentos invasivos ou que necessitem da colaboração do usuário e a ambientes reais não controlados onde as imagens são geralmente de baixa qualidade, baixa resolução e ruidosas. Por isso, os sistemas biométricos aplicados, por exemplo, na área de vigilância, devem ser os mais discretos possíveis, evitando a interação direta com o usuário e resolvendo os problemas de ambientes não controlados. Diz-se que um sistema biométrico nessas condições é não cooperativo e não controlado. Dentre os traços biométricos possíveis de serem capturados com estas restrições, a face e a íris ocupam lugar de destaque. No entanto, a face é facilmente disfarçável por adereços, maquiagens e características voláteis como barbas e bigodes. A íris e a região ocular podem ser obstruídas normalmente por óculos, mas é um traço biométrico que pode ser usado mais facilmente com as restrições impostas.

Grandes esforços têm sido focalizados para melhorar o desempenho, a escala e a usabilidade dos sistemas biométricos. O programa AADHAR da Índia (UIDAI, 2019), o Apple's Touch ID, o reconhecimento de íris em dispositivos móveis da NTT DOCOMO (TODAY, 2015) e o programa de controle de fronteira dos Emirados Árabes Unidos (UAESS, 2019) são exemplos de tais tendências em aplicações comerciais e governamentais. Devido a aplicações bem sucedidas, como as citadas anteriormente, há um crescente interesse e demanda para estender os recursos de biometria para reconhecimento de longo alcance (*Long-Range-Recognition* - LRR) onde a cooperação do usuário é minimizada.

## 2.1 Íris a Distância

Há uma quantidade significativa de pesquisas em reconhecimento de íris que utilizam imagens de alta resolução capturadas em ambientes controlados e cooperativos. Os resultados destas pesquisas já se transformaram inclusive em produtos comerciais disponíveis para reconhecimento biométrico.

O primeiro trabalho que utiliza o termo "*Iris-at-a-distance* – IAAD" foi publicado por De Villar (VILLAR; IVES; MATEY, 2010). O sistema IAAD proposto é um protótipo que ilustra a viabilidade de reconhecimento de íris a uma distância de 30 metros. O sistema usa uma câmera de campo-de-visão-largo para localizar uma pessoa pela detecção da face e dos olhos. Uma câmera de campo-de-visão-estreito, acoplada a um telescópio de oito polegadas, é então apontada para o primeiro olho detectado.

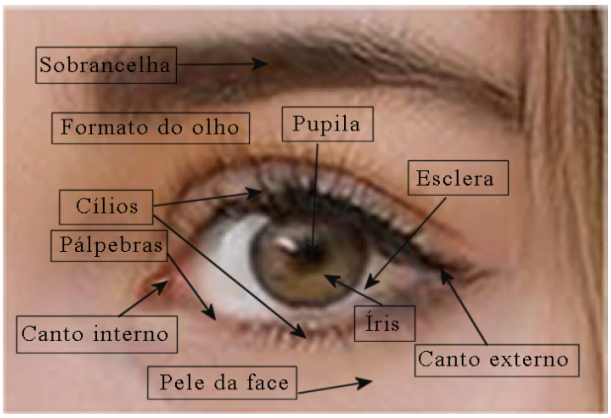
Posteriormente, o termo IAAD foi utilizado por outros pesquisadores (NGUYEN et al., 2017) para identificar o reconhecimento de íris por qualquer equipamento a distâncias superiores a 1 m sem a necessidade de colaboração do usuário.

O problema dos sistemas de IAAD é que atualmente dependem de equipamentos específicos e caros para a aquisição das imagens tais como portais com iluminação NIR (Near Infra Red) e câmeras de alta resolução espacial com lentes específicas para a situação.

## 2.2 Informação periocular

A região periocular é definida como a região facial nas imediações dos olhos, normalmente englobando as pálpebras, cílios, sobrancelhas e a área da pele vizinha. A biometria periocular analisa a forma das pálpebras, o formato do olho, a distribuições de cílios e informações de textura da pele ou esclera. O reconhecimento periocular baseia-se no senso comum da capacidade humana de "reconhecer alguém simplesmente por olhar para seus olhos", o que fornece quantidades substanciais de informação discriminante (PARK; ROSS; JAIN, 2009) (PARK et al., 2011) (AMBIKA; RADHIKA; SESHACHALAM, 2012), a qual permanece relativamente estável durante longos períodos de tempo. Os elementos típicos da região periocular são mostrados na figura 1.

Figura 1: Região Periocular




Fonte: (BARCELLOS et al., 2019)

O traço periocular é adequado para sistemas LRR, pois pode ser capturado e reconhecido em distâncias mais longas sem a necessidade de sensores adicionais. A característica biométrica periocular pode ser usada com uma resolução menor do que a necessária para a íris. Essa propriedade fornece uma grande vantagem quando a resolução ou a qualidade da íris é baixa. Como uma característica biométrica única, a região periocular fornece um bom equilíbrio entre a íris e a face em termos de distância de imagem, desempenho de reconhecimento e cooperação do usuário em condições não controladas (PARK; ROSS; JAIN, 2009) (AMBIKA; RADHIKA; SESHACHALAM, 2012) (NIGAM; VATSA; SINGH, 2015). A primeira tentativa de usar a região periocular como um traço biométrico foi realizada por Park, Ross e Jain (2009).

Os métodos tradicionais de reconhecimento periocular são baseados nas técnicas estabelecidas de visão computacional, caracterizadas por pré-processamento, segmentação, extração de características e classificação. No entanto, o fato de ser necessário processar uma imagem para extrair características e depois comparar com uma base de dados, torna os métodos tradicionais de visão computacional lentos e de difícil implementação na solução de problemas reais que requerem uma grande quantidade de imagens.

## 2.3 Biometria Ocular

Devido à eficácia do traço periocular como uma modalidade autônoma biométrica, bem como seu benefício complementar ao reconhecimento de íris, as duas modalidades podem ser também combinadas para melhorar a robustez dos sistemas LRR. A maioria das abordagens na literatura empregam Fusão por Pontuação em nível simples por soma ponderada e exibem notável melhoria em base de dados de longo alcance (TAN; KUMAR, 2012) (RAGHAVENDRA et al., 2013) (WOODARD et al., 2010) (TAN; KUMAR, 2013) (XIAO; SUN; TAN, 2012).

Santos et al (2015) utilizam uma rede neural com duas camadas escondidas treinada para fundir as pontuações obtidas pela íris e a região periocular. Enquanto a Fusão por Pontuação tem provado ser eficaz para íris e região periocular, outras abordagens como a Fusão de Características não têm sido bem exploradas. Além disso, a combinação das características de íris e periocular em ordem de importância das características obtidas pode melhorar a robustez do sistema, especialmente no caso de grandes distâncias e condições de aquisição não controladas (DOYNOV; DERAKHSHANI, 2012). A vanguarda da investigação nesta linha de pesquisa é o reconhecimento de imagens que contenham a região do olho, parte da face (periocular) e a íris. Nestas imagens, a íris pode ser de qualidade variável. Esta nova linha investigativa de pesquisa é referida como Reconhecimento Biométrico Ocular (*Biometric Ocular Recognition* – BOR) para diferenciá-lo dos problemas semelhantes de reconhecimento independente de íris e IAAD. O desafio internacional, *The Ocular Challenge* (FOCS, 2010) promovido pelo NIST (*National Institute of Standards and Technology*), objetiva encorajar o desenvolvimento de novos algoritmos nesta linha. Para isso, uma base de dados é disponibilizada publicamente com imagens de uma única região de íris e dos olhos. Estas regiões foram extraídas de sequências de vídeo capturadas em infravermelho próximo (NIR) coletadas no sistema *Iris On the Move* (IoM) (MATEY et al., 2006) e disponível no Portal MBGC (*Multiple Biometric Grand Challenge*) (MBGC, 2010). As bases disponíveis para o desafio não contemplam imagens adquiridas no espectro visível e com múltiplos sensores (*cross-sensors*) o que limita a portabilidade das soluções do desafio para reconhecimento biométrico real em ambientes não controlados e não cooperativos.

## 2.4 Reconhecimento da região periocular ou da íris usando aprendizagem profunda

Com a capacidade de aprender as características dos dados de treinamento, técnicas de aprendizagem profunda estão impulsionando o desempenho de reconhecimento de íris, especialmente sob condições não controladas.

Algumas pesquisas investigam aplicações dessa abordagem em visão computacional, especificamente para o reconhecimento de íris. Liu et al (2015) utilizam aprendizagem profunda para o Reconhecimento Heterogêno de Iris (HIR) devido à existência de grande demanda por um sistema de gerenciamento de identidade em grande escala. As imagens de íris adquiridas em ambiente heterogêneo, caracterizado pelo uso de diferentes câmeras, têm grandes variações intraclasse, tais como diferentes resoluções, diferente óptica do sensor, iluminação, escala, etc. Por métodos tradicionais, conhecidos pelo termo em inglês *handcraft*, é um desafio criar manualmente um filtro robusto para tratar as complexas variações intraclasse nas imagens de íris obtidas em meios heterogêneos. A proposta chamada de DeepIris (LIU et al., 2015) aprende características relacionais para medir a similaridade entre pares de imagens de íris, baseada em redes neurais convolucionais (CNN).

Da mesma maneira, o trabalho de De Marsico, Petrosino e Ricciardi (2016) faz um levantamento de 37 métodos de reconhecimento de íris utilizando aprendizagem de máquina, para os quais são utilizadas as bases CASIA v1, CASIA v2 e CASIA v3. Nenhum dos métodos publicadas envolve IAAD e nem utiliza aprendizagem produfnda.

Recentemente, Redes Neurais Convolucionais foram aplicadas no reconhecimento de íris utilizando a arquitetura VGG-Net (MINAEE; ABDOLRASHIDIY; WANG, 2016) nas bases CASIA-Iris-1000 e IITD, e usando a arquitetura AlexNet (ALASLANI; ELREFAEI, 2018) nas bases IITD, CASIA-Iris-V1, CASIA-Iris-1000 e CASIA-Iris-V3 Interval. Apesar dos excelentes resultados obtidos, estas bases não são adequadas para análise da região ocular, pois, as imagens foram adquiridas especificamente para reconhecimento de íris, prejudicando a visibilidade da área da região periocular. Outras arquiteturas de CNN, como a DeepIrisNet-A e DeepIrisNet-B (GANGWAR; JOSHI, 2016), foram propostas especificamente com solução para o reconhecimento da íris, sendo inviável o uso em reconhecimento ocular.

Em um trabalho recente, Lee et al. (2017) propõem um novo método de reconhecimento de imagem de íris com ruído e de imagens oculares usando uma imagem de íris e duas de regiões perioculares, baseadas em três redes neurais convolucionais (CNNs) em paralelo e fusão de pontuação. Neste trabalho a íris é detectada, segmentada, transformada em coordenadas polares e normalizada, sendo posteriormente utilizada como entrada de uma CNN. Os resultados demonstraram um acréscimo de acurácia no reconhecimento, no entanto, o fato de ter que processar as imagens de íris requer maior tempo de processa-

mento, o que pode inviabilizar o uso desta solução em aplicações que tenham uma base de dados com grande número de imagens.

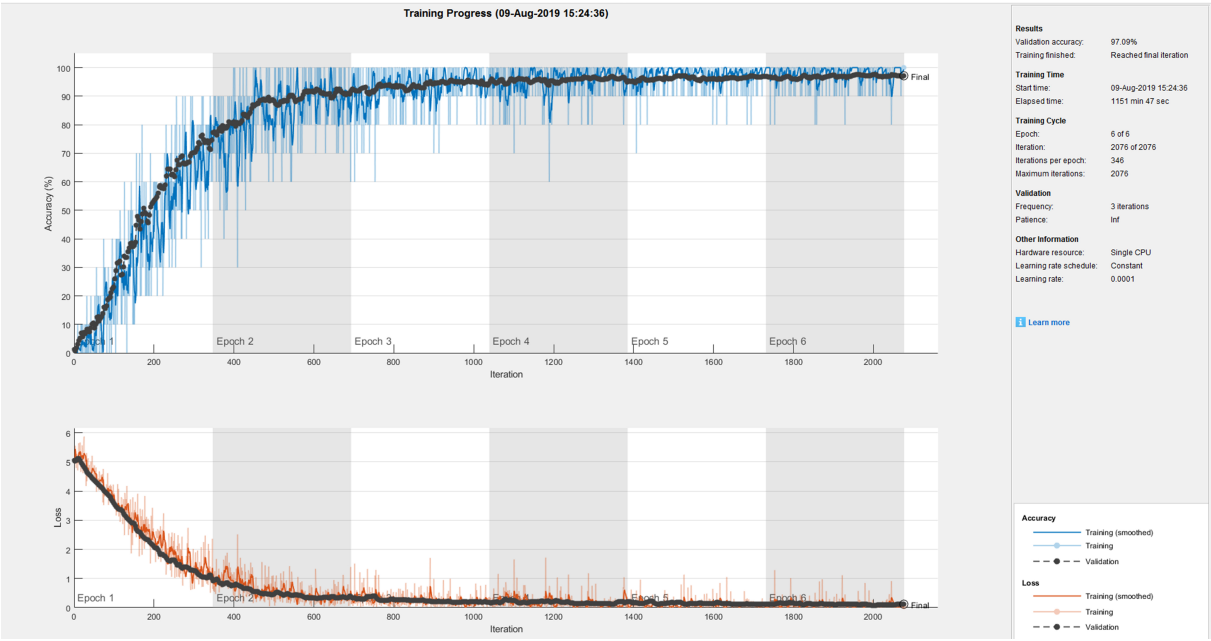
A segmentação da íris baseada em aprendizagem profunda apresenta a desvantagem de requerer um longo tempo de processamento. Para resolver esse problema, Lee et al (2019) propõem um método que encontra rapidamente uma área aproximada da íris sem segmentar com precisão a respectiva região nas imagens de entrada e reconhece a região ocular. Uma rede residual profunda (ResNet) é utilizada para resolver o problema de taxas de reconhecimento reduzidas devido ao desalinhamento entre as imagens de íris registradas e de reconhecimento. As experiências foram realizadas usando três bases de dados: CASIA-Iris-Distance, CASIA-Iris-Lamp e CASIA-Iris-Thousand (BIT, 2010). Para encontrar uma área aproximada da íris nas imagens de entrada o método proposto realiza a correspondência de modelos baseados em sub-blocos. Das três bases de dados usadas neste estudo, a base de dados CASIA-Iris-Distance possui imagens com uma grande área facial que inclui os dois olhos. Dessa forma, casos de detecção falsa, em que áreas fora do olho são detectados incorretamente podem ocorrer durante a detecção ocular que usa apenas o modelo com base em sub-blocos. Para resolver esse problema, o estudo utiliza o detector de olhos Adaboost em uma região de pesquisa que inclui os dois olhos na imagem de entrada. As ROIs da pupila e da ocular são encontradas nessa região através da correspondência de modelos baseados em sub-blocos. O aumento dos dados também foi utilizado para aumentar o número de imagens de treinamento. Foram executadas a translação de seis pixels e o recorte nas direções para cima, para baixo, para a esquerda e para a direita para aumentar os dados por um fator de 169. A base de dados Iris-Distance passou de 2567 imagens para 351520 com 142 classses. Nos testes, os modelos ResNet-50, 101 e 152 foram utilizados para realizar o ajuste fino através dos dados de treinamento aumentados. As acurácias obtidas foram de 97,42% para a ResNet-50, de 97,86% para a ResNet-101 e de 97,89% para a ResNet-152.

Dada a característica da base CASIA-Iris-Distance de ser composta por faces inteiras adquiridas à distância e em alta resolução, decidiu-se por utilizá-la neste trabalho de conclusão de curso para avaliar o desempenho de redes neurais convolucionais aplicadas ao reconhecimento biométrico da região periocular utilizando *transfer learning*.

# 3 REDES NEURAIS

Aprendizagem de máquina é o estudo de algoritmos que são capazes de se aprimorarem sozinhos. Uma das formas mais comuns de aprendizagem de máquina são as redes neurais, uma forma de algoritmo que recebe este nome pela sua semelhança estrutural com a rede de neurônios do sistema nervoso. O algoritmo, ao ser criado, tem uma acurácia muito baixa devido a sua natureza inicial aleatória, mas após várias fases de treinamento a partir de exemplos, o algoritmo aprende e pode atingir níveis de acerto surpreendentemente altos. A figura 2 mostra a crescente na acurácia de um algoritmo de rede neural conforme ele passa pelos vários ciclos de treinamento

Figura 2: Acurácia e erro de uma rede durante seu treinamento




Fonte: Autor (2019)

Uma rede neural representa um jeito novo de pensar a resolução de problemas utilizando código e linguagem de programação. Ao escrever um algoritmo convencional, o programador tem em mente tudo que ele fará; todos os passos são explicitados e o programador tem completa ciência do que ocorre em seu código linha a linha. Por outro lado, com as redes neurais, o programador é capaz de entender o código de aprendizado, mas não é capaz de entender o algoritmo já treinado.

## 3.1 Aplicações de redes neurais

As redes neurais são capazes de dar respostas a problemas complexos demais para algoritmos convencionais, como diagnósticos de doenças (NATURE, 2019) e agrupamento de dados (GOGGIN, 2018), devido à sua natureza de aprendizado. Entretanto, para um problema simples, a rede neural pode dar resultados menos precisos que um algoritmo comum. Desta forma é importante reconhecer suas limitações e em quais problemas aplicá-las.

As redes neurais se destacam em problemas em que coletam um conjunto de dados quantificáveis de entrada e dão uma resposta a partir disso, seja uma resposta numérica ou uma classificação. Um exemplo de resposta numérica é a precificação de uma casa; os dados de entrada podem envolver o número de janelas, quartos e área total e a saída seria o preço da casa. Em contraponto, um exemplo de classificação é o diagnóstico de doenças; os dados de entrada são uma lista dos sintomas e a saída uma lista de prováveis doenças.

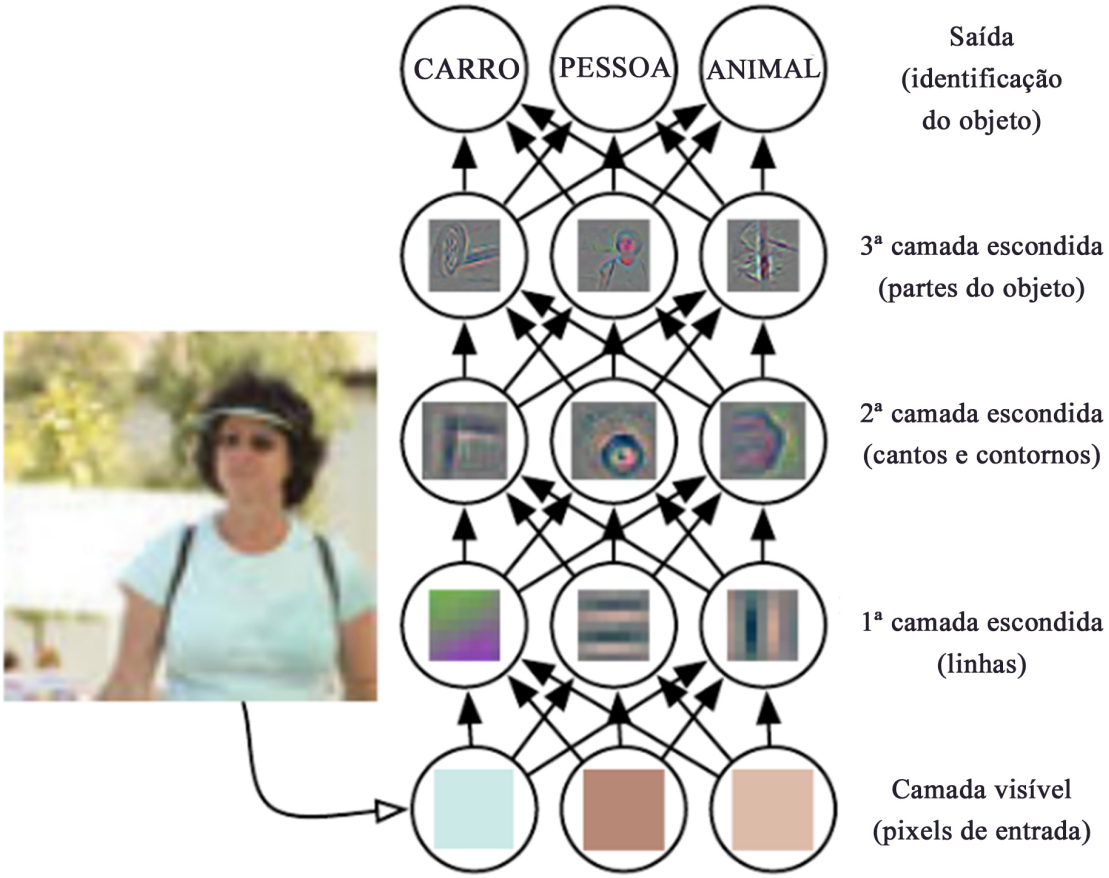
Seja pra dar uma resposta única ou uma classificação, uma rede neural precisa ser treinada exaustivamente. Sua acurácia e desempenho estão diretamente ligados à quantidade de exemplos dados, à qualidade dos exemplos e à duração do treinamento. As redes mais importantes atualmente tiveram acesso a centenas de milhares de exemplos para atingirem seus resultados (TSANG, 2018), assim deve-se escolher os problemas, e grupo de exemplos, de acordo.

Para o exemplo da precificação de casas, o algoritmo gerado pode apresentar uma boa qualidade se seu espaço amostral for pequeno, como um bairro ou uma cidade pequena. Em contrapartida, algoritmos que utilizam redes sociais como entrada costumam ser bem eficientes pela grande quantidade de exemplos fornecidos diariamente pelos seus usuários(PENNACCHIOTTI; POPESCU, 2011) (RUTHS; PFEFFER, 2014).

## 3.2 Aprendizagem profunda

As redes neurais são arquitetadas por camadas, cada camada adicionando um nível de complexidade. Nas camadas de entrada são aprendidos os conceitos mais simples, como linhas contornos e texturas, e a cada camada adicional os conceitos da camada anterior se relacionam para criar conceitos mais complexos. A esse conceito de aprendizado por camadas, cada vez mais profundas, da-se o nome de aprendizagem profunda (GOODFELLOW; BENGIO; COURVILLE, 2016). Na figura 3 é possível ver como cada camada consegue identificar conceitos mais complexos até chegar na parte final onde ocorre a classificação.

Figura 3: Evolução na complexidade das características identificadas em cada camada




Fonte: Adaptado de (GOODFELLOW; BENGIO; COURVILLE, 2016)

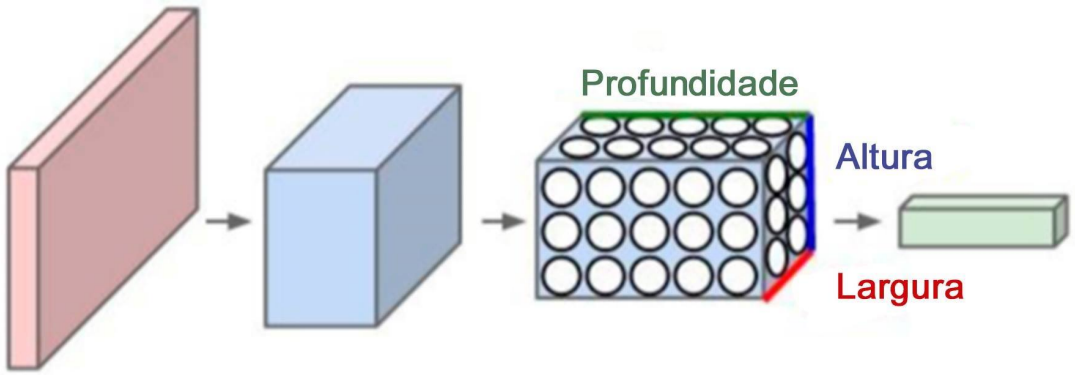
## 3.3 CNN e imagens

Redes neurais convencionais utilizam como elemento de entrada um vetor, o que causa um problema em situações onde é necessário trabalhar com imagens. Ao transformar a imagem em uma estrutura unidimensional perdem-se muitas informações sobre a vizinhança de cada pixel, parte essencial para resolução dos problemas.

Assim, para o tratamento de imagens, utilizam-se Redes Neurais Convolucionais (*Convolutional Neural Network* - CNN) (CS231N, 2016) (DESHPANDE, 2016). Estas utilizam como entrada uma matriz quadrada com um canal para imagens preto e branco e 3 canais para imagens coloridas. Desta forma, diversos problemas de visão computacional, desde reconhecimento de números e letras escritos à mão (NIELSEN, 2018) até problemas na área médica (NATURE, 2019), podem ser resolvidos com aprendizagem de máquina utilizando as CNNs.

As redes Convolucionais recebem esse nome devido aos filtros de convolução que são utilizados em suas camadas. Cada filtro trata toda a matriz, alterando e adicionando uma nova camada de informação na imagem. A convolução traz novas informações para a matriz, mas também a diminui levemente. Assim, após algumas camadas, a matriz terá reduzido em altura e largura, mas aumentado significativamente sua profundidade. A figura 4 representa o redimensionamento através das camadas.

Figura 4: Redimensionamento que acontece através das camadas




Fonte: Adaptado de (CS231N, 2016)

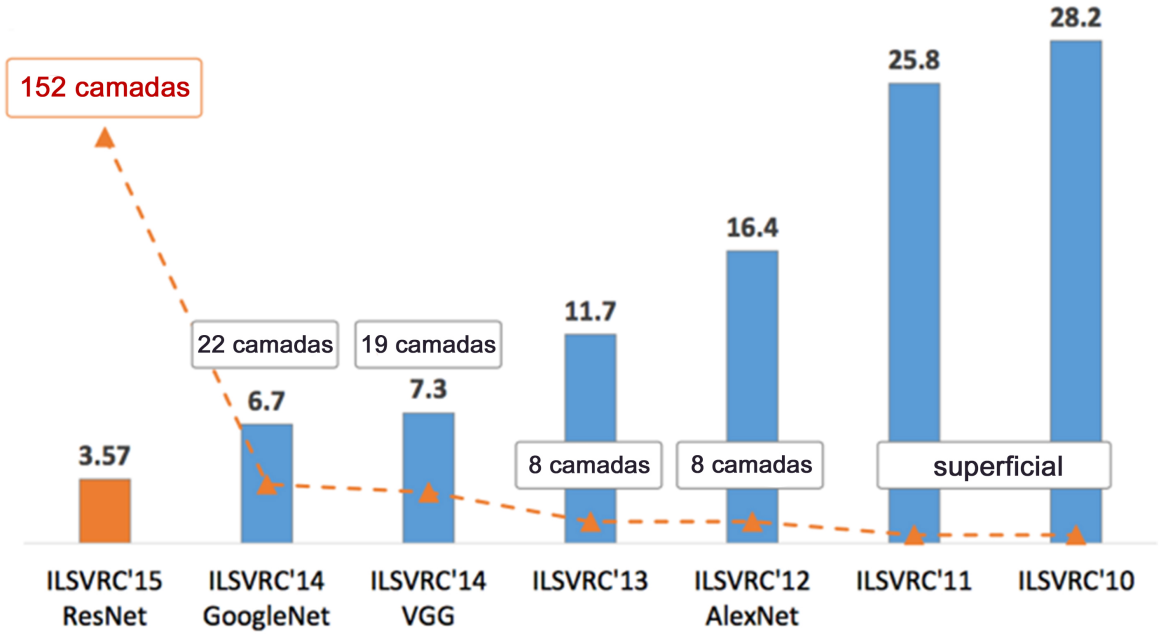
## 3.4 Camadas de uma CNN

Uma CNN é montada em camadas, cada camada recebe os dados da camada anterior e a modifica de alguma forma. A figura 5 apresenta os vencedores da *ImageNet Large Scale Visual Regocnition Challenge* (ILSVRC) (RUSSAKOVSKY et al., 2015) e o número de camadas de cada uma das redes. As camadas podem ser de vários tipos, sendo os mais comuns: entrada, filtros de convolução, *pooling*, ReLU e camada totalmente conectada.

### 3.4.1 Entrada

A primeira camada, ou camada de entrada, representa a imagem a ser analisada pelo algoritmo ainda inalterada. Essa imagem será representada por uma matriz quadrada com profundidade padronizada igual a um, para imagens em preto e branco, ou três, para imagens coloridas.

Figura 5: Vencedores do ILSVRC em cada ano, o número de camadas em suas redes e sua porcentagem de erro no desafio




Fonte: Adaptado de (DAS, 2017)

### 3.4.2 Filtros de convolução

Nessas camadas aplicam-se os filtros. Cada filtro irá analisar uma pequena parte da matriz e avançar para a parte seguinte até percorrer a matriz inteira. A relação do tamanho do filtro e o tamanho do passo é importante para que a matriz inteira seja percorrida de forma correta. Para alguns filtros é interessante que o tamanho do passo seja igual ao tamanho do filtro para que não haja setores filtrados diversas vezes. É comum também, para alguns filtros, a utilização de um *padding* de zeros na matriz para melhorar sua performance. O *padding* de zeros irá adicionar uma camada de zeros ao redor da matriz, permitindo assim que o centro do filtro atinja as bordas da imagem. O número de filtros, o tamanho de cada filtro, a utilização de *padding* e o tamanho do passo ficam todos a cargo do programador que escrever o algoritmo e alterar esses valores altera a rede neural como um todo. Ao final dessa camada, produz-se uma matriz de tamanho $a \times a \times c \times n$, sendo n o número de filtros aplicados, c o número de canais da imagem e a é dada pela equação 3.1

$$a = ((input - tam + 2 \times pad)/pas) + 1 \tag{3.1}$$

- *input* - Tamanho da matriz de entrada

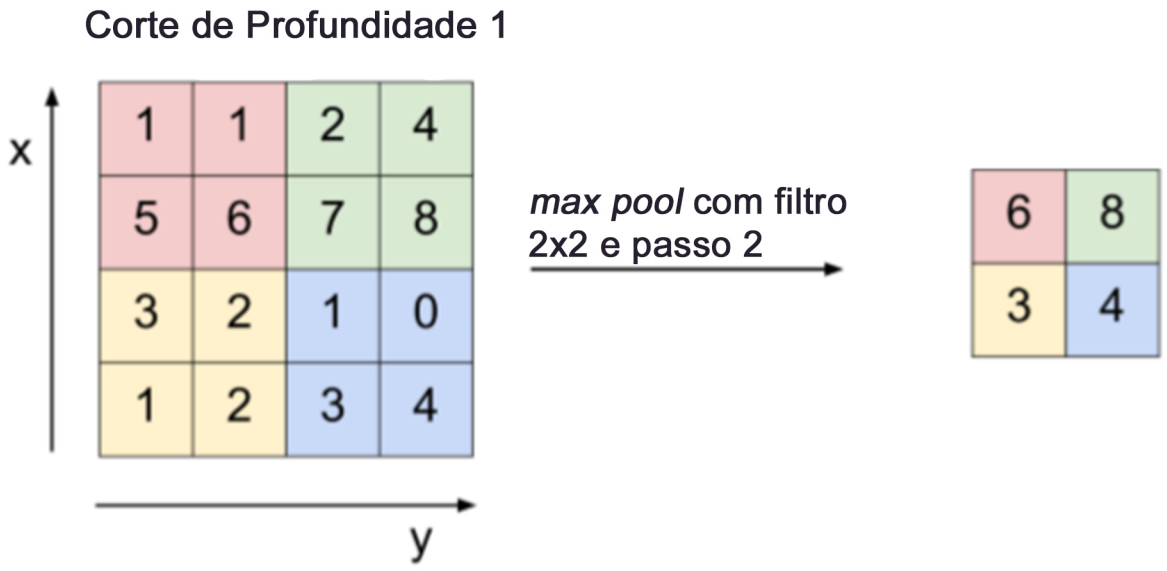
- *tam* - Tamanho dos filtros
- *pad* - Tamanho do *padding* de zeros
- *pas* - Tamanho dos passos

É possível observar pela equação 3.1 que a cada filtro a matriz ganha mais uma camada de profundidade enquanto diminui um pouco em altura e largura.

### 3.4.3 *Pooling*

Camadas de *pooling* são camadas de subamostragem utilizadas com o único propósito de diminuir a matriz de entrada. Assim, os dados recebidos são simplificados para que a próxima camada possa analisá-los de forma mais eficiente. Uma camada de *pooling* resulta em uma altura e largura menor, sem adição de informação, portanto a profundidade se mantém a mesma. A figura 6 mostra um exemplo de *max pooling*. É possível ver, através da figura, como a camada de *pooling* diminui a matriz de entrada. Não há um método de *pooling* que se destaque, apenas alguns mais utilizados pela sua simplicidade. Além do *max pooling*, também é comum o uso do *average pooling*, onde se utiliza a média dos valores da região em vez do seu valor máximo.

Figura 6: Representação de uma camada de *max pooling*




Fonte: Adaptado de (CS231N, 2016)

### 3.4.4 ReLU

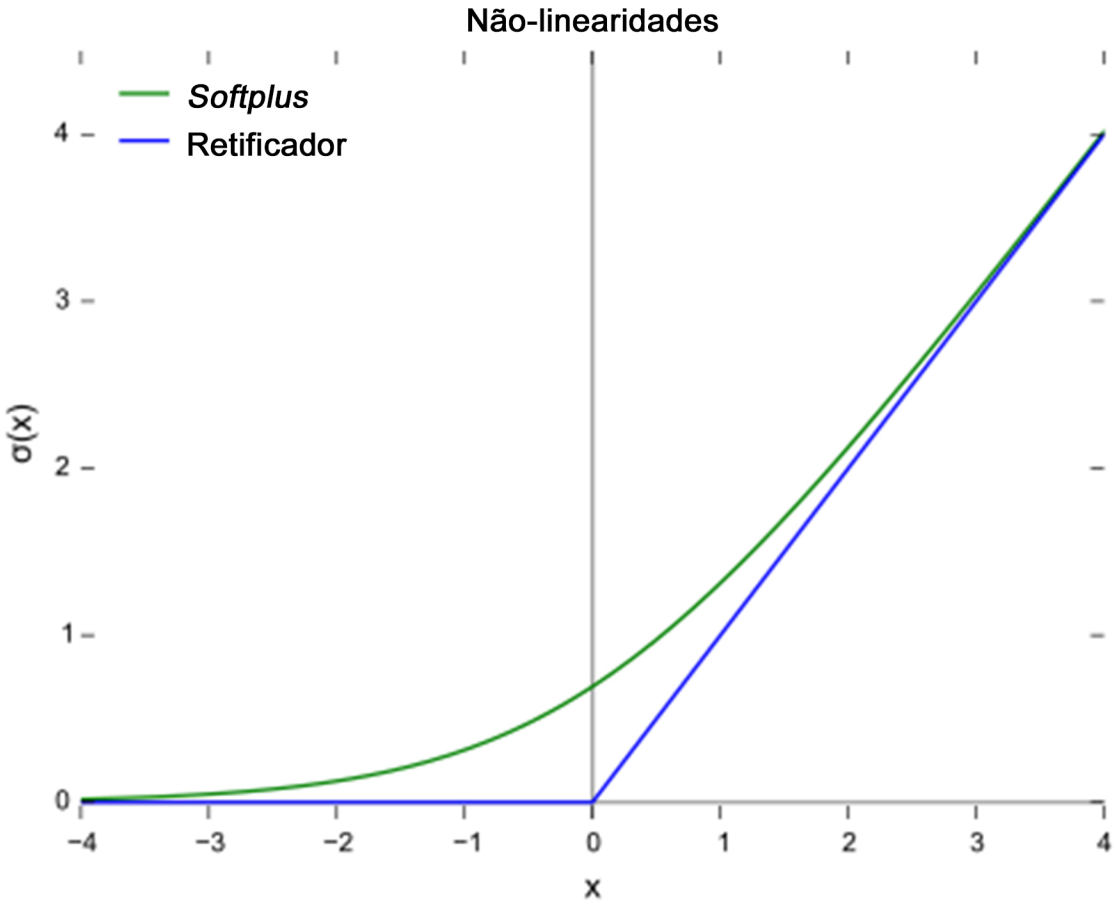
ReLU significa *Rectified Linear Unit* e representa a camada de retificação linear que costuma ser aplicada após camadas de convolução.

A imagem de entrada pode ser interpretada como um vetor e os filtros como outro vetor, tornando o processo de convolução em um produto escalar onde a resposta é um vetor que mapeia a similaridade entre os trechos da imagem original e o filtro.

Utilizando uma função não linear como $f(x) = max(0, X)$ pode-se destacar melhor os pontos importantes da imagem. Outra grande vantagem é substituir os grandes valores negativos fora do esperado para o trecho por zeros, o que facilita o cálculo e poupa processamento computacional, ao custo de negligenciar parte da imagem que seja interessante no momento.

A função de máximo é apenas um exemplo simples, mas diversas funções não lineares são utilizadas como camadas de ReLU. A figura 7 mostra a função softplus, uma das mais utilizadas, em comparação com a função $max(0, X)$.

Figura 7: Gráfico mostrando a diferença entre uma função $max(0, X)$ e softplus.




Fonte: Adaptado de (ETER, 2016)

### 3.4.5 Camada Totalmente Conectada

Conforme a matriz vai passando por camadas de filtro e camadas de *pooling*, suas dimensões se alteram. A cada camada de filtro, a matriz ganha mais profundidade, contudo perde altura e largura a depender do filtro utilizado. Com as camadas de *pooling*, as dimensões diminuem ainda mais resultando em uma matriz bem pequena e com várias camadas de profundidade. Nesse ponto da rede, começam as camadas totalmente conectadas (*Fully connected layers* - FCL).

Nas FCL, a CNN se comporta como uma rede neural convencional e portanto precisa de um vetor de entrada em vez de uma matriz. Desta forma, a matriz, que já está adequada para tal, é transforada num vetor de tamanho igual ao produto das dimensões da matriz de entrada. Exemplificando, se na entrada havia uma imagem de dimensões $224 \times 224 \times 3$ e após várias camadas de *pooling* e filtros ela se tornou uma matriz $7 \times 7 \times 3 \times 28$ então o vetor de entrada da FCL será de 4116 elementos.

A diferença entre as camadas do tipo filtro de convolução e totalmente conectadas é que nas primeira os dados só são relacionados com aqueles suficientemente próximos para serem abordados pelo filtro ao mesmo tempo. Em contrapartida, nas FCL os dados são inter relacionados por completo. Portanto, nas camadas convolutivas há um aprendizado de quais características são importantes e a cada camada mais profunda, geram-se características mais complexas. Enquanto isso, nas FCL essas características já estabelecidas são analisadas e relacionadas para preparar a rede para uma classificação.

## 3.5 Treinamento

### 3.5.1 *Forward propagation e backpropagation*

Quando inicializada a rede neural, todos os seus parâmetros começam com valores aleatórios que serão ajustados para cada problema durante a fase de treinamento. Esse treinamento da rede neural se dá através de duas etapas: *forward propagation* e *backpropagation*.

Durante a fase de *forward propagation*, a imagem passa pelas camadas com parâmetros aleatórios e chega a um resultado final, que será comparado a um gabarito para medir a eficiência da rede neural. Com isso define-se a função perda (também conhecida como função erro ou *loss*), onde as variáveis são os valores e pesos dos filtros em cada camada de ativação e a resposta é o quão distante do gabarito a rede neural está.

Após estabelecer a função erro, é possível utilizar derivadas para encontrar o menor erro possível. Para a função *loss*, não seria viável encontrar seus valores de mínimo local, uma vez que a mesma possui muitas variáveis, porém pode-se calcular o gradiente descendente da função. Deste modo, mesmo sem a informação exata do local do mínimo erro, conhece-se a direção em que o erro diminui. Assim, na fase de *backpropagation*, o

algoritmo ajusta as variáveis das camadas apropriadas dando um passo na direção certa, de modo que o erro da rede neural seja menor.

Após o término do *backpropagation* e com um algoritmo um pouco modificado, retorna-se a fase de *forward propagation*. Esse ciclo se repete diversas vezes e a cada iteração o algoritmo torna-se capaz de aumentar o número de acertos para o qual está sendo treinado. O processo se mantém até a melhora no erro ser menor do que um limite pré estabelecido.

Existem duas constantes importantes a serem definidas pelo responsável pelo desenvolvimento do algoritmo de treinamento durante o processo: o tamanho do passo e o limite do erro. O tamanho do passo indica quanto cada variável mudará na direção do gradiente a cada iteração de *backpropagation*. Enquanto passos pequenos estenderão o tempo de aprendizado, valores grandes dificultam a precisão do algoritmo, podendo até prendê-lo em um *loop* onde o limite do erro nunca é atingido.

O limite do erro determina quando o algoritmo não consegue mais melhorar sua performance. Ao escolher um valor muito alto, perde-se precisão desnecessariamente no algoritmo, mas ao escolher um valor muito baixo é possível que o algoritmo nunca atinja o objetivo desejado.

### 3.5.2 *Batch Size* e Treinamento estocástico

O *forward propagation* e *backpropagation* se utilizam da função perda (*loss*) para determinar a acurácia da rede neural. Sabendo que é necessário aplicar o gradiente da função, ao invés de sua derivada, um problema surge: não é possível saber se o mínimo apontado pelo gradiente é o mínimo possível ou apenas um mínimo local.

Com isso em vista, é utilizada apenas uma pequena amostra da base de imagens cujo tamanho é denominado *batch size*. Com tamanhos pequenos, cada iteração do treinamento tem seu próprio gradiente, condição que contorna o problema de mínimos locais além de acelerar o processo de cálculo da função perda. Quando o treinamento é feito com *batch size* igual a um, ele é chamado de treinamento estocástico.

### 3.5.3 *Transfer Learning*

Para que os filtros deixem de ter valores aleatórios característicos da inicialização dos parâmetros da criação da rede neural, é necessária uma grande quantidade de imagens e tempo de treinamento para passarem a gerar informações úteis.

Uma vez treinada, a rede está pronta para realizar apenas a função designada, o que gera diversas situações com necessidade de treiná-la novamente, como casos em que se é preciso adicionar mais uma classe a um classificador. Por métodos clássicos, isso exige que toda a rede seja treinada novamente, o que além de demandar tempo, exige a

disponibilidade das imagens usadas no primeiro treinamento, recurso nem sempre viável.

Para resolver os problemas supracitados, surgiram as técnicas de *transfer learning* que possibilitam utilizar uma rede já treinada como base para sua nova rede. Assim, é possível adicionar novas funções para uma rede já existente em um tempo muito mais curto e sem precisar de toda a base de imagens do treinamento inicial.

Como os filtros das CNN representam formas abstratas porém universais, uma CNN pode ser pré treinada para reconhecer certos padrões mais amplos e, posteriormente, refinada para um propósito mais específico utilizando a técnica de *transfer learning*. Um exemplo é o uso de uma rede já treinada para carros na identificação de motos, visto que as características de ambos tipos de veículos são semelhantes (PISTORI, 2017).

Com isso, surgem grandes empresas gerando CNN's robustas treinadas com milhares de imagens sendo capazes de reconhecer diversos padrões diferentes. Essas CNN's são então disponibilizadas para a criação de redes com boa acurácia utilizando apenas uma fração do tempo e das imagens, como AlexNet, LeNet, ResNet e GoogLeNet.

A técnica de *transfer learning* pode ser feita de diversas formas, com cada variante se diferenciando pela forma que lida com a comparação entre os parâmetros já existentes e a serem adicionados. Os três tipos mais comuns são: *Fine Tuning*, *Feature Extraction* e *Joint Trainning* (LI; HOIEM, 2016).

O método de *Feature Extraction* considera que a rede pré treinada já extrai as características da imagem de forma coerente para o novo problema, não modificando o começo da rede mas apenas suas camadas finais, onde estão os classificadores. O treinamento com base nas camadas finais torna o processo mais rápido, requerendo poucas imagens de treino. Como boa parte da rede se mantém inalterada, sua escolha é importante para a extração das características desejadas.

Por outro lado, *Fine Tuning* modifica a rede inteira durante o treinamento. Assim como em *Feature Extraction*, alteram-se as camadas finais para as novas tarefas, porém durante o treinamento, as imagens novas são usadas para ajustar levemente também as camadas iniciais para um melhor desempenho. Desta forma, é possível realizar um treinamento focado na nova aplicação sem a necessidade de retreinar por completo as camadas iniciais. Uma cópia da rede original pode ser feita antes do *Fine Tuning*, criando-se uma rede levemente alterada, para cada nova tarefa.

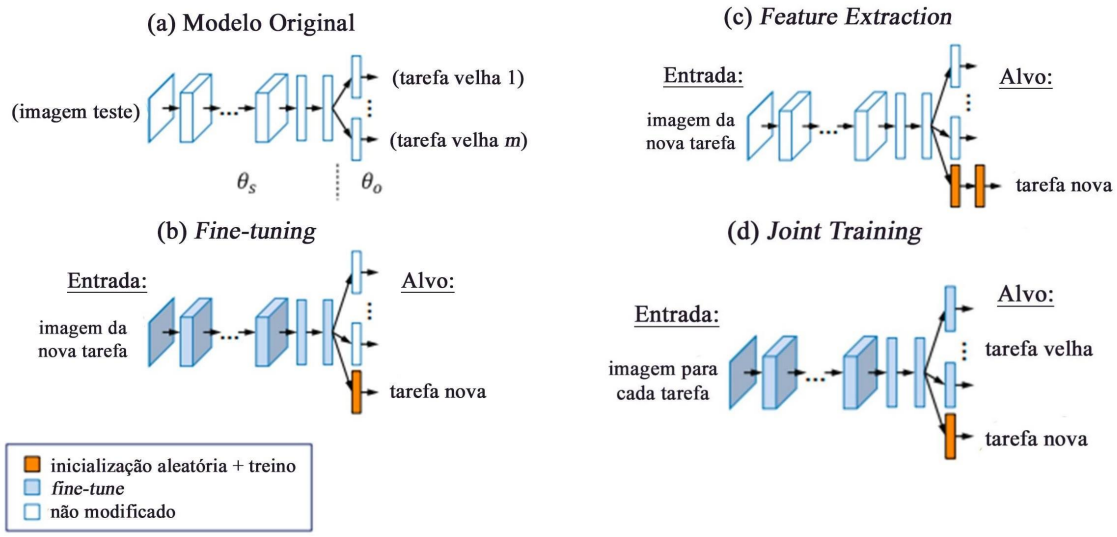
*Joint Training* é similar a *Fine Tunning*, com a diferença em que as imagens utilizadas para o treinamento da rede original serão adicionadas às imagens do novo treinamento. Assim, minimiza-se a perda de acurácia nas tarefas já existentes da rede ao implementar a nova.

No artigo Learning Without Forgetting (LI; HOIEM, 2016), os autores propõem outra técnica para *transfer learning* enquanto discorrem sobre as existentes:

> *"Each of these strategies has a major drawback. Feature extraction typically underperforms on the new task because the shared parameters fail to represent some information that is discriminative for the new task. Fine-tuning degrades performance on previously learned tasks because the shared parameters change without new guidance for the original task-specific prediction parameters. Duplicating and fine-tuning for each task results in linearly increasing test time as new tasks are added, rather than sharing computation for shared parameters. Joint training becomes increasingly cumbersome in training as more tasks are learned and is not possible if the training data for previously learned tasks is unavailable." (LI; HOIEM, 2016)*

A figura 8 adaptada de (LI; HOIEM, 2016) ilustra as diferenças entre as técnicas de *transfer learning* e as comparam com a técnica que propõem.

Figura 8: Diferentes técnicas de *transfer learning*.




Fonte: Adaptado de (LI; HOIEM, 2016)

## 3.6 CNNs pré-treinadas

Uma das partes mais importantes em *transfer learning*, é escolher uma boa CNN de base. Nessa sessão serão introduzidas algumas CNN reconhecidas no meio acadêmico pelas suas capacidades e por serem marcos no desenvolvimento de CNN. Os resultados dessas redes são anualmente validados na ILSVRC (RUSSAKOVSKY et al., 2015).

### 3.6.1 AlexNet

AlexNet foi a primeira CNN a ser aplicada a reconhecimento de imagens em larga escala. Foi proposta por Alex Krizhevsky (KRIZHEVSKY; SUTSKEVER; HINTON, 2015) e trazia soluções novas para problemas de *overfitting*, vencendo assim o ILSVRC de 2012. Seu treinamento levou de 5 a 6 dias em duas GPUs NVIDIA GTX 580 3GB e utilizou mais de 1 milhão de imagens. Sua arquitetura inclui uma camada de entrada, cinco camadas de convolução com ReLU, três camadas de *pooling* e três camadas completamente conectadas totalizando doze camadas.

### 3.6.2 GoogLeNet

GoogLeNet foi proposta em (SZEGEDY et al., 2014) e venceu o ILSVRC de 2014. Possui uma arquitetura de vinte duas camadas e troca as diversas camadas completamente conectadas no final por uma camada de *pooling* e uma única camada completamente conectada. Essa troca é reconhecida como uma boa alternativa por salvar mais parâmetros vindo das camadas convolucionais. A GoogLeNet também introduziu o conceito de *Inception module* onde a saída de uma camada passa por diversas camadas diferentes em paralelo para no fim seus resultados serem recombinados.

### 3.6.3 ResNet

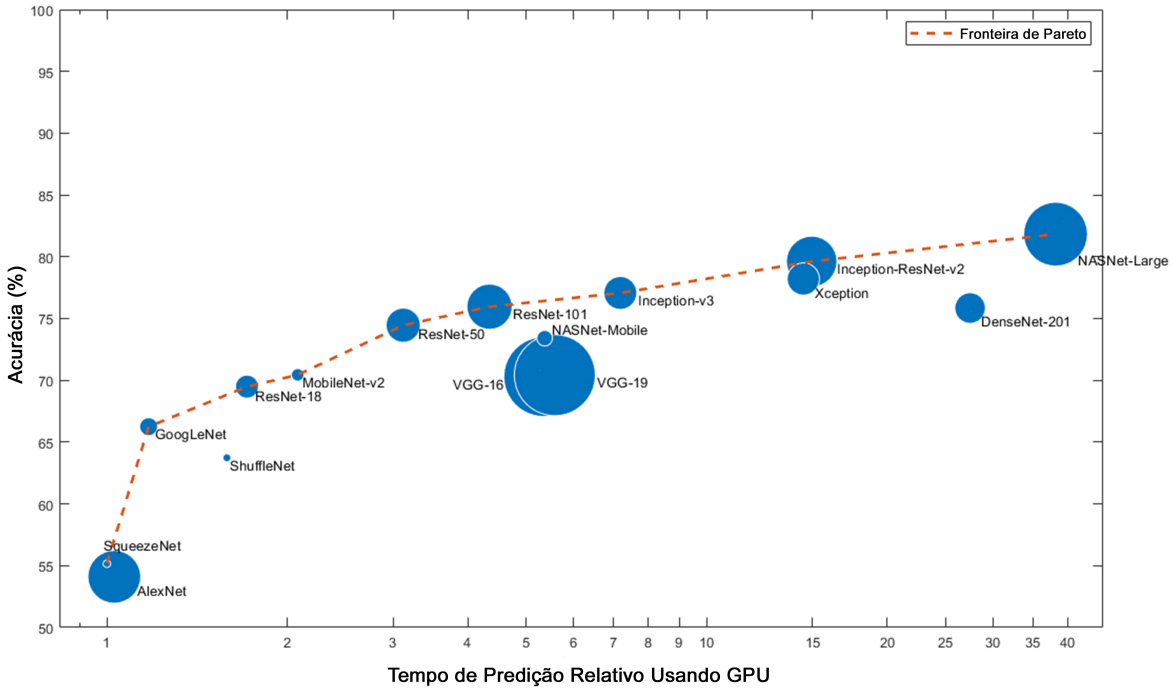
A ResNet foi proposta em (HE et al., 2015) e venceu o ILSVRC de 2015. O nome *Residual Network* (ResNet) vem de uma técnica de blocos residuais que permitem a ResNet vencer o problema da degradação por camada e assim fazer redes com grande número de camadas. Desta forma, a ResNet foi a primeira rede bem sucedida com mais de 100 camadas. ResNet possui quatro versões: ResNet18, ResNet50, ResNet101 e ResNet152, tendo respectivamente dezoito, cinquenta, cento e uma e cento e cinquenta e duas camadas.

# 4 MATERIAL E MÉTODO

## 4.1 Escolha das CNNs

Para a execução de todos os treinamento e testes, foi utilizado o software MATLAB versão R2018a em uma CPU Intel i7-6700. As redes pré treinadas disponíveis neste programa são mostradas na figura 9 pela sua classificação quanto à acurácia e tempo de predição. Para este trabalho, foram escolhidas as redes AlexNet, GoogLeNet, ResNet18 e ResNet50 por apresentarem baixo tempo de predição e por terem funções já implementadas na biblioteca do MATLAB.

Figura 9: Redes disponíveis no software MATLAB quanto à acurácia e tempo de predição.




Fonte: Adaptado de (MATHWORKS, 2018a)

Em cada rede, foram retiradas as camadas finais de *softmax*, FCL e classificação e recriadas camadas novas condizentes com o problema a ser abordado. Nos apêndices B, C, D e E estão presentes os algoritmos utilizados para fazer o *transfer learning* nas redes e a validação em cada tipo de imagem.

A técnica de *transfer learning* escolhida foi a de *Fine Tunning* com uma taxa de aprendizado de 20 para 1, ou seja, as mudanças nas novas camadas aconteceriam com um peso 20 vezes maior do que para as camadas já existentes. O valor da proporção foi

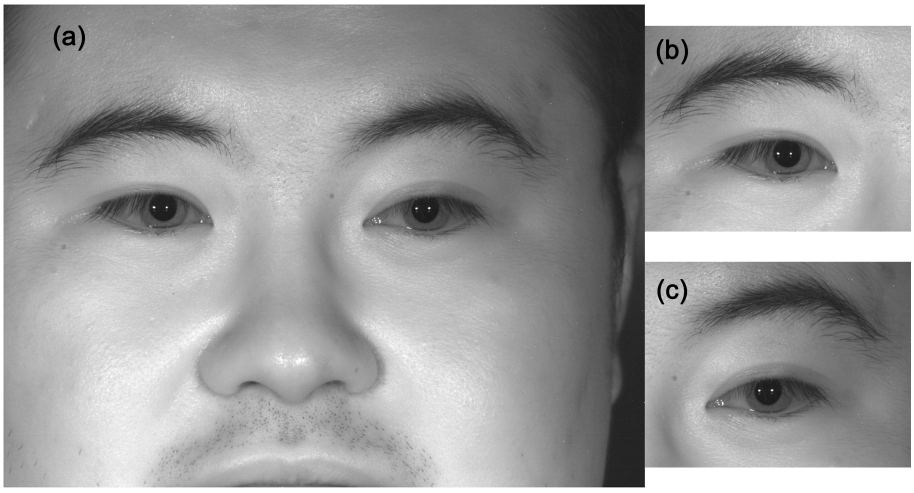
sugerida por (MATHWORKS, 2018b). Não foi feito nenhum tipo de *data augmentation* nas imagens pois o objetivo do trabalho é conseguir bons resultados com um limitado número de imagens. Assim, apenas um redimensionamento foi feito para atender as dimensões de entrada de cada rede.

## 4.2 Base de Imagens utilizada

A base de imagens utilizada nos treinamentos é o subgrupo CASIA-Iris-Distance da base CASIA-IrisV4 (BIT, 2010). A base CASIA-IrisV4 é um grupo de imagens utilizada para biometria a partir da iris do individuo coletadas pela *Chinese Academy of Sciences' Institute of Automation* (CASIA). No subgrupo CASIA-Iris-Distance utilizado, as fotos são tiradas a distância, com alta resolução e do rosto inteiro. São um total de 2567 fotos divididas em 142 classes, uma para cada individuo. As fotos foram tiradas a distâncias de 3 metros com uma resolução de $2352 \times 1728$ pixels.

As imagens utilizadas neste trabalho foram recortadas manualmente na região periocular dos indivíduos por meio do software Photoshop, criando 3 tipos de imagens: fotos do rosto inteiro, fotos apenas da região periocular esquerda, e fotos da região periocular direita. Na figura 10 é possível ver um exemplo de uma imagem original e suas regiões perioculares recortadas.

Figura 10: Exemplo da imagem de um individuo,(a) e de suas regiões perioculares direita,(b) e esquerda,(c) recortadas




Fonte: (a) (BIT, 2010), (b) Autor (2019) e (c) Autor (2019)

Para a análise de desempenho das CNNs, é necessário dividir as imagens em três grupos: 70% das imagens formam um subconjunto de treinamento, 15% formam um subconjunto de validação e 15% um subconjunto de teste. As imagens foram divididas de forma aleatória com o uso do algoritmo que está no apêndice A. O algoritmo ainda

criou outros dois tipos de imagem: a região periocular esquerda ou direita invertida no eixo vertical. Assim, passam-se a ter os seguintes tipos de imagens para os testes:

1. Rosto Inteiro
2. Região Periocular Esquerda
3. Região Periocular Direita
4. Região Periocular Esquerda Invertida
5. Região Periocular Direita Invertida
6. Ambas as Regiões Perioculares

## 4.3 Experimentos

Cada uma das quatro redes foram treinadas quatro vezes separadas, totalizando dezesseis experimentos, e em cada treinamento foi analisada a acurácia em cada tipo de imagem. Foram então anotados os tempos de treinamento e as acurácias para cada experimento.

Primeiramente, cada algoritmo foi treinado com ambas as regiões perioculares (tipo 6) e então foi analisada a sua capacidade de acerto com os tipos de imagem de regiões perioculares (tipos 2 a 6). Em seguida, o treinamento foi feito com as regiões perioculares esquerda e direita (tipos 2 e 3) e entre cada treinamento foram testadas as capacidades de acerto de forma similar ao primeiro treinamento. Por fim, realizou-se um treinamento com imagens do rosto inteiro (tipo 1), validado apenas com imagens também do rosto inteiro, para uma analise final comparativa entre o reconhecimento da região periocular e o reconhecimento facial.

# 5 RESULTADOS

Os experimentos foram realizados utilizando quatro redes, AlexNet, GoogLeNet, ResNet18 e ResNet50 e cada rede foi treinada quatro vezes. A primeira com imagens do rosto inteiro e testada apenas com imagens do rosto inteiro, para que uma comparação entre região periocular e a face inteira possa ser feita. O segundo treinamento foi feito com imagens da região periocular esquerda e direita juntas em um mesmo grupo, para este ensaio foram usados todos os tipos de imagem de região periocular para fazer os testes. O terceiro treinamento foi feito usando imagens da região periocular esquerda e o quarto treinamento com imagens da região periocular direita, em ambos os casos, foram usadas todos os tipos de imagens da região periocular para se fazer os testes.

## 5.1 AlexNet

A AlexNet teve os resultados mais rápidos, porém os menos precisos, tal como esperado pela figura 9. As tabelas 1 a 4 mostram os resultados dos experimentos.

Para o primeiro experimento, treinado e testado com imagens do rosto inteiro, a AlexNet obteve os seguintes resultados:

Tabela 1: Acurácia da rede AlexNet treinada com o rosto inteiro

| **AlexNet - Rosto inteiro** | |
|---|---|
| Tempo de treinamento | 115m 08s |
| Acurácia rosto inteiro | 0,9355 |

Fonte: Autor (2019)

Para o segundo experimento, treinado com imagens de ambas as regiões perioculares, a Alexnet obteve os seguintes resultados:

Tabela 2: Acurácias da rede AlexNet treinada com ambas regiões perioculares

| **AlexNet - Ambas regiões perioculares** | |
|---|---|
| Tempo de treinamento | 284m 01s |
| Acurácia ambos | 0,9171 |
| Acurácia dir. | **0,9215** |
| Acurácia esq. | 0,9128 |
| Acurácia dir. inv. | 0,8508 |
| Acurácia esq. inv. | 0,8385 |

Fonte: Autor (2019)

Para o terceiro experimento, treinado com imagens de da região periocular esquerda, a Alexnet obteve os seguintes resultados:

Tabela 3: Acurácias da rede AlexNet treinada com a região periocular esquerda

| **AlexNet - Região periocular esquerda** | |
|---|---|
| Tempo de treinamento | 82m 48s |
| Acurácia ambos | 0,6593 |
| Acurácia dir. | 0,4136 |
| Acurácia esq. | **0,9000** |
| Acurácia dir. inv. | 0,4581 |
| Acurácia esq. inv. | 0,8282 |

Fonte: Autor (2019)

Para o quarto experimento, treinado com imagens de da região periocular direita, a Alexnet obteve os seguintes resultados:

Tabela 4: Acurácias da rede AlexNet treinada com a região periocular direita

| **AlexNet - Região periocular direita** | |
|---|---|
| Tempo de treinamento | 79m 35s |
| Acurácia ambos | 0,6386 |
| Acurácia dir. | **0,9005** |
| Acurácia esq. | 0,3821 |
| Acurácia dir. inv. | 0,7958 |
| Acurácia esq. inv. | 0,4154 |

Fonte: Autor (2019)

## 5.2 GoogLeNet

A GoogLeNet teve resultados melhores que a AlexNet, porém os tempos de treinamento foram por volta de 2.5 vezes maiores. As tabelas 5 a 8 mostram os resultados dos experimentos.

Para o primeiro experimento, treinado e testado com imagens do rosto inteiro, a GoogLeNet obteve os seguintes resultados:

Tabela 5: Acurácia da rede GoogLeNet treinada com o rosto inteiro

| **GoogLeNet - Rosto inteiro** | |
|---|---|
| Tempo de treinamento | 226m 38s |
| Acurácia rosto inteiro | 0,9306 |

Fonte: Autor (2019)

Para o segundo experimento, treinado com imagens de ambas as regiões perioculares, a GoogLeNet obteve os seguintes resultados:

Tabela 6: Acurácias da rede GoogLeNet treinada com ambas regiões perioculares

| **GoogLeNet - Ambas regiões perioculares** | |
|---|---|
| Tempo de treinamento | 779m 37s |
| Acurácia ambos | 0,9288 |
| Acurácia dir. | **0,9293** |
| Acurácia esq. | 0,9282 |
| Acurácia dir. inv. | 0,8979 |
| Acurácia esq. inv. | 0,9077 |

Fonte: Autor (2019)

Para o terceiro experimento, treinado com imagens de da região periocular esquerda, a GoogLeNet obteve os seguintes resultados:

Tabela 7: Acurácias da rede GoogLeNet treinada com a região periocular esquerda

| **GoogLeNet - Região periocular esquerda** | |
|---|---|
| Tempo de treinamento | 219m 34s |
| Acurácia ambos | 0,6775 |
| Acurácia dir. | 0,4450 |
| Acurácia esq. | **0,9051** |
| Acurácia dir. inv. | 0,4948 |
| Acurácia esq. inv. | 0,8923 |

Fonte: Autor (2019)

Para o quarto experimento, treinado com imagens de da região periocular direita, a GoogLeNet obteve os seguintes resultados:

Tabela 8: Acurácias da rede GoogLeNet treinada com a região periocular direita

| **GoogLeNet - Região periocular direita** | |
|---|---|
| Tempo de treinamento | 207m 42s |
| Acurácia ambos | 0,6503 |
| Acurácia dir. | **0,9162** |
| Acurácia esq. | 0,3897 |
| Acurácia dir. inv. | 0,8194 |
| Acurácia esq. inv. | 0,4410 |

Fonte: Autor (2019)

## 5.3 ResNet18

A ResNet18 obteve resultados mais precisos que a GoogLeNet com tempos de treinamento menores, diferente do esperado pela figura 9, se mostrando mais eficiente. As tabelas 9 a 12 mostram os resultados dos experimentos.

Para o primeiro experimento, treinado e testado com imagens do rosto inteiro, a ResNet18 obteve os seguintes resultados:

Tabela 9: Acurácia da rede ResNet18 treinada com o rosto inteiro

| **ResNet18 - Rosto inteiro** | |
|---|---|
| Tempo de treinamento | 197m 56s |
| Acurácia rosto inteiro | 0,9578 |

Fonte: Autor (2019)

Para o segundo experimento, treinado com imagens de ambas as regiões perioculares, a ResNet18 obteve os seguintes resultados:

Tabela 10: Acurácias da rede ResNet18 treinada com ambas regiões perioculares

| **ResNet18 - Ambas regiões perioculares** | |
|---|---|
| Tempo de treinamento | 667m 25s |
| Acurácia ambos | 0,9469 |
| Acurácia dir. | **0,9476** |
| Acurácia esq. | 0,9462 |
| Acurácia dir. inv. | 0,9136 |
| Acurácia esq. inv. | 0,9154 |

Fonte: Autor (2019)

Para o terceiro experimento, treinado com imagens de da região periocular esquerda, a ResNet18 obteve os seguintes resultados:

Tabela 11: Acurácias da rede ResNet18 treinada com a região periocular esquerda

| **ResNet18 - Região periocular esquerda** | |
|---|---|
| Tempo de treinamento | 191m 11s |
| Acurácia ambos | 0,6982 |
| Acurácia dir. | 0,4555 |
| Acurácia esq. | **0,9359** |
| Acurácia dir. inv. | 0,5288 |
| Acurácia esq. inv. | 0,8231 |

Fonte: Autor (2019)

Para o quarto experimento, treinado com imagens de da região periocular direita, a ResNet18 obteve os seguintes resultados:

Tabela 12: Acurácias da rede ResNet18 treinada com a região periocular direita

| **ResNet18 - Região periocular direita** | |
|---|---|
| Tempo de treinamento | 180m 13s |
| Acurácia ambos | 0,6710 |
| Acurácia dir. | **0,9293** |
| Acurácia esq. | 0,4179 |
| Acurácia dir. inv. | 0,8246 |
| Acurácia esq. inv. | 0,5385 |

Fonte: Autor (2019)

## 5.4 ResNet50

A ResNet50 obteve os resultados mais precisos, porem os mais lentos, conforme esperado pela figura 9. Similar a comparação entre AlexNet e GoogLeNet, a ResNet18 e ResNet50 apresentam acurácias próximas, mas com tempos de treinamento bem distintos com a ResNet50 chegando a tempos 3,5 vezes maiores. As tabelas 13 a 16 mostram os resultados dos experimentos.

Para o primeiro experimento, treinado e testado com imagens do rosto inteiro, a ResNet50 obteve os seguintes resultados:

Tabela 13: Acurácia da rede ResNet50 treinada com o rosto inteiro

| **ResNet50 - Rosto inteiro** | |
|---|---|
| Tempo de treinamento | 584m 29s |
| Acurácia rosto inteiro | 0,9529 |

Fonte: Autor (2019)

Para o segundo experimento, treinado com imagens de ambas as regiões perioculares, a ResNet50 obteve os seguintes resultados:

Tabela 14: Acurácias da rede ResNet50 treinada com ambas regiões perioculares

| **ResNet50 - Ambas regiões perioculares** | |
|---|---|
| Tempo de treinamento | 2303m 37s |
| Acurácia ambos | 0,9482 |
| Acurácia dir. | 0,9450 |
| Acurácia esq. | **0,9513** |
| Acurácia dir. inv. | 0,9162 |
| Acurácia esq. inv. | 0,9333 |

Fonte: Autor (2019)

Para o terceiro experimento, treinado com imagens de da região periocular esquerda, a ResNet50 obteve os seguintes resultados:

Tabela 15: Acurácias da rede ResNet50 treinada com a região periocular esquerda

| **ResNet50 - Região periocular esquerda** | |
|---|---|
| Tempo de treinamento | 659m 47s |
| Acurácia ambos | 0,6671 |
| Acurácia dir. | 0,4005 |
| Acurácia esq. | **0,9282** |
| Acurácia dir. inv. | 0,4921 |
| Acurácia esq. inv. | 0,8821 |

Fonte: Autor (2019)

Para o quarto experimento, treinado com imagens de da região periocular direita, a ResNet50 obteve os seguintes resultados:

Tabela 16: Acurácias da rede ResNet50 treinada com a região periocular direita

| **ResNet50 - Região periocular direita** | |
|---|---|
| Tempo de treinamento | 613m 24s |
| Acurácia ambos | 0,7150 |
| Acurácia dir. | **0,9476** |
| Acurácia esq. | 0,4872 |
| Acurácia dir. inv. | 0,9162 |
| Acurácia esq. inv. | 0,5103 |

Fonte: Autor (2019)

## 5.5 Análise dos resultados

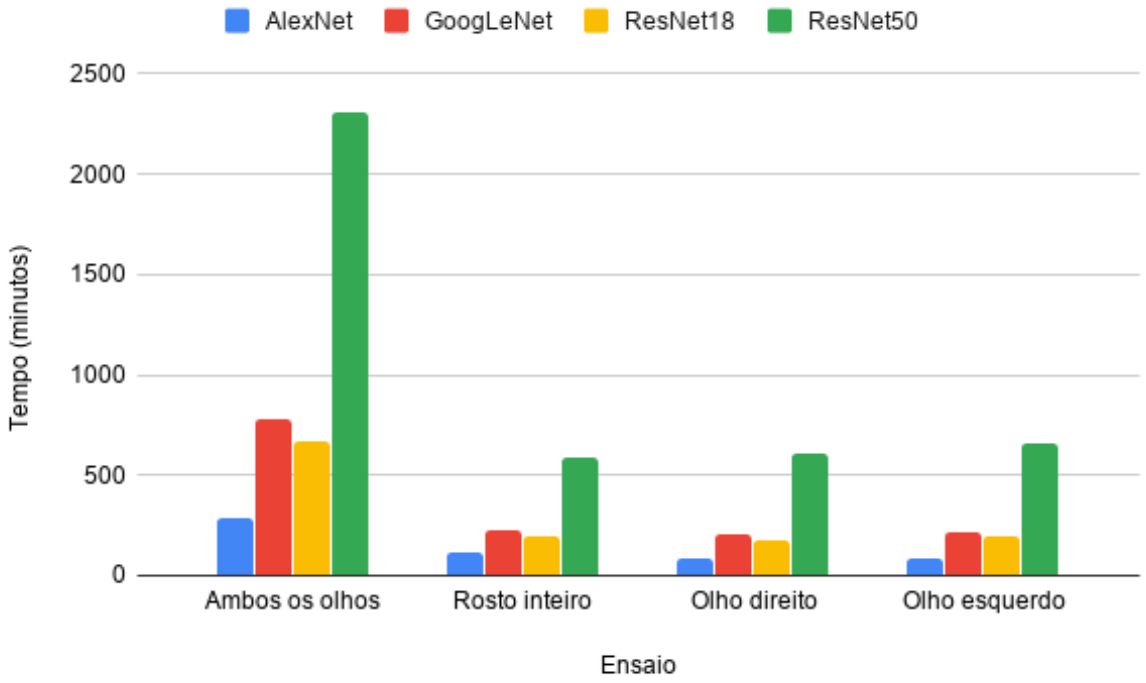
### 5.5.1 *Transfer Learning*

A primeira análise realizada sobre a eficácia da técnica de *transfer learning* no reconhecimento biométrico da região periocular mostra que: todas as redes obtiveram níveis de acurácia iguais ou superiores a noventa por cento quando validadas com o mesmo tipo de imagem que foram treinadas e o treinamento de todas as redes durou algumas horas com tempos variando de 1h19m35s a 38h23m37s. Considerando o número de imagens limitado e os baixos tempos de treino, a técnica de *transfer learning* apresenta acurácias altas mostrando assim um alto desempenho.

### 5.5.2 Tempos de treinamento

No gráfico da figura 11 são apresentados os tempos de treinamento de cada rede. Fica evidente uma relação entre a complexidade da rede e o tempo necessário para o seu treinamento, com as redes com mais camadas sempre apresentando resultados mais lentos. A ResNet18 ainda apresenta resultados mais rápidos que a GoogLeNet, contrariando o esperado pela figura 9, mas reforçando a ideia de que menos camadas aceleram o processo de treinamento. Outro ponto a ser ressaltado é a mudança que ocorre nos treinamentos com tipos diferentes de imagem. Imagens da região periocular esquerda ou direita possuem tempos de treino similares pois possuem conjuntos de teste com o mesmo número de amostras.

Figura 11: Tempo de processamento na etapa de treinamento para as quatro redes consideradas, por ensaio




Fonte: Autor (2019)

Ao juntar as duas regiões perioculares em um mesmo subconjunto de treinamento, a quantidade de dados dobra, aumentando o tempo deste processo. Pode-se também observar que as imagens de rosto inteiro obtiveram um tempo de treinamento muito próximo àquelas das regiões perioculares recortadas. Observa-se assim que a maior complexidade nas imagens do rosto inteiro não eleva o tempo de treinamento e que aumentar o número de imagens no subconjunto de treinamento tem um impacto muito mais significativo.

### 5.5.3 Região periocular

Os resultados mostram que as acurácias dos experimentos feitos com o rosto inteiro são maiores que as acurácias dos experimentos feitos com regiões perioculares. Entretanto, comparando os resultados para redes treinadas e testadas com o mesmo tipo de imagem, percebe-se que a diferença de acurácias é pequena. A tabela 17 mostra as diferenças entre acurácias do rosto inteiro e acurácias de regiões perioculares, treinadas e testadas com o mesmo tipo de imagem, em cada rede.

Tabela 17: Diferenças de acurácia da região periocular e do rosto inteiro

| | Região periocular: | | | |
|---|---|---|---|---|
| | Esquerda | Direita | Ambas | Médias |
| AlexNet | 0,0355 | 0,0350 | 0,0184 | 0,0296 |
| GoogLeNet | 0,0255 | 0,0144 | 0,0018 | 0,0139 |
| ResNet18 | 0,0219 | 0,0285 | 0,0109 | 0,0204 |
| ResNet50 | 0,0247 | 0,0053 | 0,0047 | 0,0116 |

Fonte: Autor (2019)

Os resultados mostram que as médias das diferenças de acurácias ficam entre 1% e 3%. Considerando os tempos de treinamento similares mostrados na figura 11 e as baixas diferenças de acurácia evidencias na tabela 17 é possível concluir que não há uma perda significativa na escolha da região periocular ao invés do rosto inteiro. Assim, a escolha de uma região ou outra depende de outros fatores como facilidade de disfarce e estabilidade da região.

### 5.5.4 Olhos invertidos

As imagens das regiões perioculares esquerda e direita foram invertidas e utilizadas na avaliação de desempenho das redes. Isso foi adotado porque, nas técnicas de *image augmentation* utilizadas para aumentar a quantidade de imagens de treinamento de CNNs, a inversão (*flip*) de imagens é uma das opções. Desta forma, pretende-se avaliar se é possível usar apenas imagens de uma das regiões perioculares e aumentar os dados para se fazer a identificação de ambas. Para que esta forma de aumento de dados seja possível, é necessário que ambas as regiões perioculares tenham características similares, porem invertidas. Assim, a intenção deste trabalho foi avaliar a similaridade das características dos olhos opostos, verificando se é possível utilizar um olho para o treinamento e o olho oposto invertido no teste. O Tipo de Teste da figura 12 pode ser detalhado como:

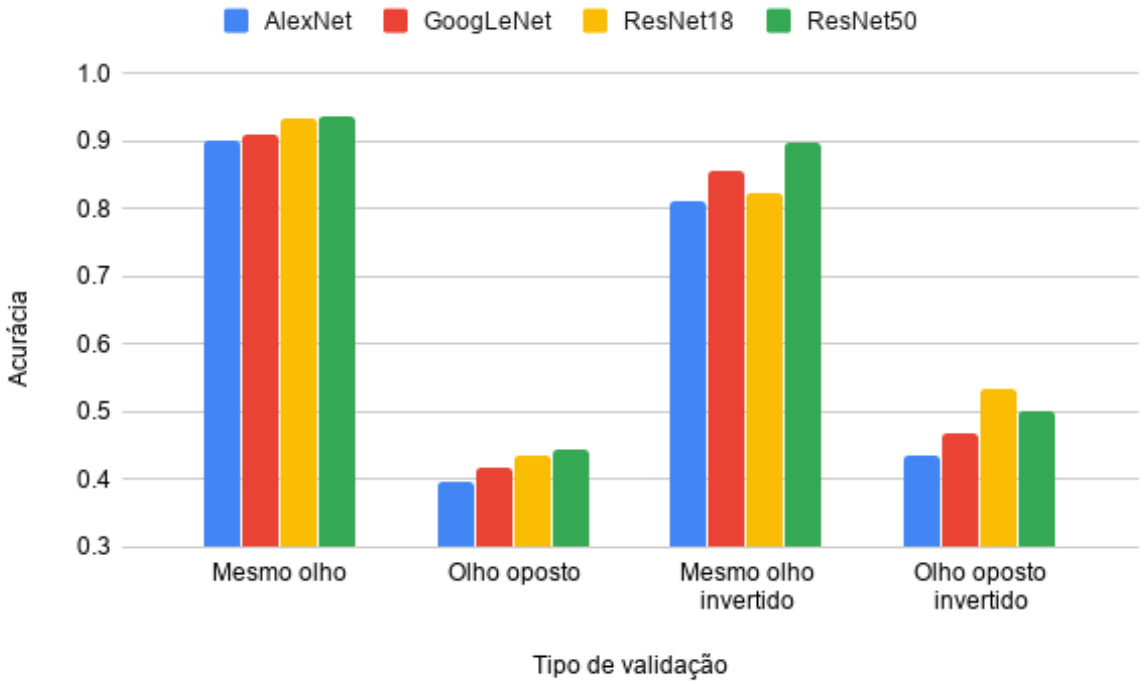
- Mesmo Olho: médias das acurácias obtidas pelas 4 redes avaliadas quando o treinamento e o teste são realizados com imagens de olhos da mesma região da face, ou seja, treinamento e teste com apenas imagens de olhos esquerdos e treinamento e teste com apenas imagens de olhos direitos.

- Olho Oposto: médias das acurácias obtidas pelas 4 redes avaliadas quando o treinamento e o teste são realizados com imagens de olhos de regiões opostas da face, ou seja, treinamento com apenas imagens de olhos esquerdos e teste com imagens de olhos direitos, e treinamento com apenas imagens de olhos direitos e teste com imagens de olhos esquerdos.

- Mesmo Olho Invertido: médias das acurácias obtidas pelas 4 redes avaliadas quando o treinamento é realizado com imagens de olhos de uma região da face e o teste é realizado com imagens de olhos desta mesma região mas invertido por software, ou seja, treinamento com apenas imagens de olhos esquerdos e teste com imagens de olhos esquerdos invertidos e treinamento com imagens de olhos direitos e teste com apenas imagens de olhos direitos invertidos (*flip*).
- Olho Oposto Invertido: médias das acurácias obtidas pelas 4 redes avaliadas quando o treinamento é realizado com imagens de olhos de uma região da face e o teste é realizado com imagens de olhos da outra região mas invertido por software, ou seja, treinamento com apenas imagens de olhos esquerdos e teste com imagens de olhos direitos invertidos e treinamento com imagens de olhos direitos e teste com apenas imagens de olhos esquerdos invertidos (*flip*).

Para montar o gráfico da figura 12 foram feitas as médias para cada item acima. Dessa forma o item "mesmo olho"terá o valor obtido do treinamento e teste de imagens do olho esquerdo somado ou valor do treinamento e teste de imagens do olho direito, divido por dois. Neste gráfico é possível observar que os resultados mostram baixos níveis de acerto, entre 38% e 55%, em todas as redes quando esta é treinada com imagens de um olho e o teste é realizado com imagens do outro olho invertido. Resultados ainda mostraram níveis de acurácia média entre 79% e 89% para treinamento com um olho e teste com o mesmo olho invertido.

Figura 12: Acurácia média, por CNN, relativa ao mesmo olho, olho oposto, mesmo olho invertido e olho oposto invertido




Fonte: Autor (2019)

Estes resultados mostram que as características de cada olho são individuais e vão além da rotação da imagem. Este fato fica evidente nos altos valores de acerto para classificação do mesmo olho, porém invertido, e dos baixos valores de acerto para o olho oposto, também invertido. Os resultados mostram ainda valores levemente mais altos de acerto com resultados obtidos com o olho oposto invertido em relação ao olho oposto sem inversão. Isso mostra que, apesar de um olho esquerdo invertido "parecer"um olho direito, a rede não o classifica corretamente quando for treinada com olhos de apenas uma região da face, e receber na fase de teste olhos da outra região da face, invertido ou não. Testar olhos invertidos em redes treinadas com olhos diretos causa apenas um pequeno ganho de acurácia em relação ao teste direto com olhos opostos esquerdo/direito ou direito/esquerdo.

# 6 CONCLUSÃO

Reconhecimento biométrico de pessoas é uma área de estudo que vem se desenvolvendo muito por conta da crescente busca por formas de identificação mais seguras. Recentemente, as Redes Neurais Convolucionais (Convolutional Neural Networks – CNN) surgiram como o novo estado-da-arte para reconhecimento de padrões, exemplificado por resultados notáveis na classificação de imagens, detecção e segmentação. A chave para seu sucesso é a capacidade de alavancar grandes conjuntos de dados rotulados para aprender cada vez mais transformações complexas da entrada e capturar invariâncias. No entanto, em algumas aplicações, como no reconhecimento biométrico, a quantidade de imagens rotuladas necessárias para treinar uma CNN "do zero", nem sempre está disponível. As CNNs treinadas em grandes conjuntos de dados geram extratores de características polivalentes e transferíveis para outros domínios. A transferência de domínio em CNNs geralmente é conseguida utilizando-se como características a saída de uma camada da rede profunda e totalmente conectada. Com o uso de *transfer learning*, CNNs podem ser rapidamente implementadas e facilmente adaptadas para aplicações em outros domínios, como no caso de reconhecimento biométrico periocular.

O objetivo geral deste trabalho foi o de avaliar o desempenho de CNNs previamente treinadas em grandes bases de dados, fazendo-se um ajuste fino para o reconhecimento da região periocular de faces humanas, utilizando-se a técnica de *transfer learning*. Como no caso de bases de dados pequenas, prevê-se sempre o uso do aumento de dados visando melhorar o desempenho da rede, os experimentos propostos neste trabalho visaram avaliar também se a inversão (*flip*) de imagens de olhos de uma região da face (esquerda ou direita) pode ser usada com boa acurácia no reconhecimento de olhos de outra região da face (direita ou esquerda). Os resultados mostraram que olho esquerdo (ou direito) invertido não é reconhecido como seu oposto, ou seja, se uma rede foi treinada com periocular esquerda a CNN falha em reconhecer perioculares direitas invertidas, e vice-versa. No entanto, as CNNs avaliadas demonstraram bom desempenho de reconhecimento quando um olho invertido é testado em uma rede treinada com olhos do mesmo lado da face. Ou seja, as redes acertam mais de 90% das imagens quando treinadas com ambos os olhos e sem inversão. Mas, acertam entre 80% e 90% das imagens quando treinadas com perioculares do mesmo lado das imagens de teste invertidas, ou seja, reconhece bem olho direito (ou esquerdo) invertido se treinada com imagens do mesmo tipo sem inversão.

Foi também analisado o desempenho da utilização da região periocular para se fazer a biometria, comparando redes treinadas com rosto inteiro e com a região periocular. Os resultados evidenciam que usar a região periocular não diminui significativamente a acurácia das redes, com perdas médias entre 1% e 3%, mostrando que esta tem características

discriminativas suficientes para o reconhecimento biométrico. Considerando vantagens como a estabilidade da região e facilidade de aquisição de imagens, conclui-se que a região periocular é uma alternativa viável ao uso da face inteira.

## 6.1 Sugestões para trabalhos futuros

Parte deste trabalho se dedicou a analise de algoritmos treinados com imagens de um olho, e testadas com o olho oposto invertido. Apesar dos valores obtidos mostrarem que cada olho possui características discriminativas distintas, ainda não foi descartado completamente a hipótese de uma melhora no desempenho da CNN utilizando aumento de dados com inversão das imagens. Desta forma, trabalhos futuros poderiam analisar mais a fundo como as diversas formas de aumento de dados interferem no reconhecimento da região periocular.

# REFERÊNCIAS


ALASLANI, M. G.; ELREFAEI, L. A. Convolutional neural network based feature extraction for iris recognition. **International Journal of Computer Science Information Technology (IJCSIT)**, v. 10, n. 2, p. 65–78, April 2018.

AMBIKA, D.; RADHIKA, K.; SESHACHALAM, D. The eye says it all: periocular region methodologies. **International Conference on Multimedia Computing and Systems (ICMCS)**, p. 180–185, 2012.

BARCELLOS, W. et al. Evaluation of fine tuning and feature extraction methods in biometric periocular recognition. **WORKSHOP DE VISÃO COMPUTACIONAL (WVC)**, v. 15, p. 43–48, sep 2019.

BIT. **Biometrics Ideal Test**. 2010. <http://biometrics.idealtest.org/dbDetailForUser.do?id=4>. (Accessed on 10/09/2019).

CS231N. **CS231n Convolutional Neural Networks for Visual Recognition**. 2016. <http://cs231n.github.io/convolutional-networks/#fc>. (Accessed on 10/09/2019).

DAS, S. **CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more....** 2017. <https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5>. (Accessed on 10/09/2019).

DELAC, K.; GRGIC, M. A survey of biometric recognition methods. In: **Proceedings. Elmar-2004. 46th International Symposium on Electronics in Marine**. [S.l.: s.n.], 2004. p. 184–193.

DESHPANDE, A. **A Beginner's Guide To Understanding Convolutional Neural Networks – Adit Deshpande – Engineering at Forward | UCLA CS '19**. 2016. <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>. (Accessed on 10/09/2019).

DOYNOV, P.; DERAKHSHANI, R. A standoff system for noncooperative ocular biometrics. In: **2012 IEEE Conference on Technologies for Homeland Security (HST)**. [S.l.: s.n.], 2012. p. 144–149.

ETER, A. B. **What are the advantages of ReLU over softmax in deep neural network?** 2016. <https://www.quora.com/What-are-the-advantages-of-ReLU-over-softmax-in-deep-neural-network>. (Accessed on 10/09/2019).

FOCS. **Face and Ocular Challenge Series (FOCS) | NIST**. 2010. <https://www.nist.gov/programs-projects/face-and-ocular-challenge-series-focs>. (Accessed on 10/22/2019).

GANGWAR, A.; JOSHI, A. Deepirisnet: Deep iris representation with applications in iris recognition and cross-sensor iris recognition. In: **2016 IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2016. p. 2301–2305.

GOGGIN, M. **Machine Learning in Construction: How Clustering Data Can Improve Processes (part 2 of 2) | Enstoa**. 2018. <https://enstoa.com/blog/machine-learning-construction-how-clustering-data-can-improve-processes-part-2-of-2>. (Accessed on 10/09/2019).

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

HE, K. et al. Deep residual learning for image recognition. **CoRR**, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

_____. **ImageNet Classification with Deep Convolutional Neural Networks**. 2015. <http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghee.pdf>. (Accessed on 10/09/2019).

LI, Z.; HOIEM, D. Learning without forgetting. **CoRR**, abs/1606.09282, 2016. Disponível em: <http://arxiv.org/abs/1606.09282>.

LIU, N. et al. Deepiris: Learning pairwise filter bank for heterogeneous iris verification. **Pattern Recognition Letters**, v. 82, 10 2015.

MATEY, J. R. et al. Iris on the move: Acquisition of images for iris recognition in less constrained environments. **Proceedings of the IEEE**, v. 94, n. 11, p. 1936–1947, Nov 2006.

MATHWORKS. **Pretrained Deep Neural Networks - MATLAB & Simulink**. 2018. <https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html>. (Accessed on 10/09/2019).

_____. **Transfer Learning Using AlexNet - MATLAB & Simulink**. 2018. <https://www.mathworks.com/help/deeplearning/examples/transfer-learning-using-alexnet.html>. (Accessed on 10/09/2019).

MBGC. **Multiple Biometric Grand Challenge (MBGC) | NIST**. 2010. <https://www.nist.gov/programs-projects/multiple-biometric-grand-challenge-mbgc>. (Accessed on 10/22/2019).

MINAEE, S.; ABDOLRASHIDIY, A.; WANG, Y. An experimental study of deep convolutional features for iris recognition. In: **2016 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)**. [S.l.: s.n.], 2016. p. 1–6.

NATURE. Ascent of machine learning in medicine. **Nature Materials**, v. 18, n. 5, p. 407–407, 2019. ISSN 1476-4660. Disponível em: <https://www.nature.com/articles/s41563-019-0360-1>.

NGUYEN, K. et al. Long range iris recognition: A survey. **Pattern Recognition**, v. 72, 05 2017.

NIELSEN, M. A. misc, **Neural Networks and Deep Learning**. Determination Press, 2018. Disponível em: <http://neuralnetworksanddeeplearning.com/>.

NIGAM, I.; VATSA, M.; SINGH, R. Ocular biometrics: A survey of modalities and fusion approaches. **Information Fusion**, v. 26, p. 1–35, 2015. ISSN 1566-2535. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1566253515000354>.

PARK, U. et al. Periocular biometrics in the visible spectrum. **IEEE Transactions on Information Forensics and Security**, v. 6, n. 1, p. 96–106, March 2011.

PARK, U.; ROSS, A.; JAIN, A. K. Periocular biometrics in the visible spectrum: A feasibility study. In: **2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems**. [S.l.: s.n.], 2009. p. 1–6.

PENNACCHIOTTI, M.; POPESCU, A.-M. A machine learning approach to twitter user classification. In: **ICWSM**. [S.l.: s.n.], 2011.

PISTORI, H. **Deep Learning - Redes Neurais Convolucionais - YouTube**. 2017. <https://www.youtube.com/watch?v=DXnyuUZcAAI>. (Accessed on 10/09/2019).

RAGHAVENDRA, R. et al. Combining iris and periocular recognition using light field camera. In: **2013 2nd IAPR Asian Conference on Pattern Recognition**. [S.l.: s.n.], 2013. p. 155–159.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

RUTHS, D.; PFEFFER, J. Social media for large studies of behaviour. **Science**, v. 346, p. 1063–4, 11 2014.

SAEZ-TRIGUEROS, D.; MENG, L.; HARTNETT, M. Face recognition: From traditional to deep learning methods. **CoRR**, abs/1811.00116, 2018. Disponível em: <http://arxiv.org/abs/1811.00116>.

SZEGEDY, C. et al. Going deeper with convolutions. **CoRR**, abs/1409.4842, 2014. Disponível em: <http://arxiv.org/abs/1409.4842>.

TAN, C.; KUMAR, A. Human identification from at-a-distance images by simultaneously exploiting iris and periocular features. In: **Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)**. [S.l.: s.n.], 2012. p. 553–556.

_____. Towards online iris and periocular recognition under relaxed imaging constraints. **IEEE Transactions on Image Processing**, v. 22, n. 10, p. 3751–3765, Oct 2013.

TODAY, B. T. Ntt docomo implements fingerprint and iris biometrics. **Biometric Technology Today**, v. 2015, n. 6, p. 2 – 3, 2015. ISSN 0969-4765. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0969476515300886>.

TSANG, S.-H. **Review: AlexNet, CaffeNet — Winner of ILSVRC 2012 (Image Classification)**. 2018. <https://medium.com/coinmonks/paper-review-of-alexnet-caffenet-winner-in-ilsvrc-2012-image-classification-b93598314160>. (Accessed on 10/09/2019).

UAESS. **United Arab Emirates - Safety & Security | export.gov**. 2019. <https://www.export.gov/article?id=United-Arab-Emirates-Safety-Security>. (Accessed on 10/22/2019).

UIDAI. **Home - Unique Identification Authority of India | Government of India**. 2019. <https://uidai.gov.in/>. (Accessed on 10/22/2019).

VILLAR, J. A. D.; IVES, R. W.; MATEY, J. R. Design and implementation of a long range iris recognition system. In: **2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers**. [S.l.: s.n.], 2010. p. 1770–1773.

WOODARD, D. L. et al. On the fusion of periocular and iris biometrics in non-ideal imagery. In: **2010 20th International Conference on Pattern Recognition**. [S.l.: s.n.], 2010. p. 201–204.

XIAO, L.; SUN, Z.; TAN, T. Fusion of iris and periocular biometrics for cross-sensor identification. In: . [S.l.: s.n.], 2012. p. 202–209.

# Apêndices

# APÊNDICE A – ALGORITMO PARA DIVISÃO DOS GRUPOS

```
function flipanddiv701515
clear all;


Dir = dir('C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais\Esq');
NumCat = sum([Dir(~ismember({Dir.name},{'.','..'})).isdir])

for i = 0:NumCat - 1
    i
    iPasta = sprintf('%d',i);
    if i<100
        iPasta = sprintf('0%d',i);
    end
    if i<10
        iPasta = sprintf('00%d',i);
    end
    str = ['C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais\Esq\' iPasta];
    Dir = dir([ str '/*.jpg']);
    NumFotos = numel(Dir);
    jlim70 = round(0.7*NumFotos) - 1;
    jlim85 = round(0.85*NumFotos) - 1;
    for j = 0:NumFotos - 1

        iFoto = sprintf('%d',j);

        clear flipft;
        clear Data;

        strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais\Esq\' iPasta '\p' iPasta 'f' iFoto 'E.jpg' ];

        Data = imread(strft);
        flipft = flip(Data,2);

```

```
34          if j<jlim70
35
36              strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                    Olhos\3Canais70\Esq\' iPasta '\p' iPasta 'f'
                    iFoto 'E.jpg'];
37              imwrite(Data,strft);
38
39              strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                    Olhos\3Canais70\Esqflip\' iPasta '\p' iPasta 'f'
                    iFoto 'E.jpg'];
40              imwrite(flipft,strft);
41
42              strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                    Olhos\3Canais70\Ambos\' iPasta '\p' iPasta 'f'
                    iFoto 'E.jpg'];
43              imwrite(Data,strft);
44
45          elseif j>jlim85
46
47              strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                    Olhos\3Canais15v\Esq\' iPasta '\p' iPasta 'f'
                    iFoto 'E.jpg'];
48              imwrite(Data,strft);
49
50              strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                    Olhos\3Canais15v\Esqflip\' iPasta '\p' iPasta 'f'
                     iFoto 'E.jpg'];
51              imwrite(flipft,strft);
52
53              strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                    Olhos\3Canais15v\Ambos\' iPasta '\p' iPasta 'f'
                    iFoto 'E.jpg'];
54              imwrite(Data,strft);
55          else
56
57
58              strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                    Olhos\3Canais15t\Esq\' iPasta '\p' iPasta 'f'
                    iFoto 'E.jpg'];
```

```
59                imwrite(Data,strft);
60
61                strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                      Olhos\3Canais15t\Esqflip\' iPasta '\p' iPasta 'f'
                       iFoto 'E.jpg'];
62                imwrite(flipft,strft);
63
64                strft = ['C:\Users\Convidado\Desktop\Lahiri\Base\
                      Olhos\3Canais15t\Ambos\' iPasta '\p' iPasta 'f'
                      iFoto 'E.jpg'];
65                imwrite(Data,strft);
66
67            end
68
69        end
70    end
71
72    end
```

# APÊNDICE B – ALGORITMO DE *TRANSFER LEARNING* DA ALEXNET

```
function AlexNet

%Carregando as imagens

imdsTrain = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais70\Ambos','IncludeSubfolders',true,'LabelSource','foldernames');
imdsTrainof = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais15t\Ambos','IncludeSubfolders',true,'LabelSource','foldernames');

imdsValidationA = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais15v\Ambos','IncludeSubfolders',true,'LabelSource','foldernames');
imdsValidationE = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais15v\Esq','IncludeSubfolders',true,'LabelSource','foldernames');
imdsValidationD = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais15v\Dir','IncludeSubfolders',true,'LabelSource','foldernames');
imdsValidationEf = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais15v\Esqflip','IncludeSubfolders',true,'LabelSource','foldernames');
imdsValidationDf = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\Base\Olhos\3Canais15v\Dirflip','IncludeSubfolders',true,'LabelSource','foldernames');


%Carregando a AlexNet
net = alexnet;

%Parametros para a rede
inputSize = net.Layers(1).InputSize;
numClasses = numel(categories(imdsTrain.Labels));

%Adaptacao da AlexNet para o problema escolhido
```

```
23 layersaux = net.Layers(1:end-3);
24 layers = [
25     layersaux
26     fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'
       BiasLearnRateFactor',20)
27     softmaxLayer
28     classificationLayer];
29
30 %Redimensionamento das imagens
31
32 augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain,
    'ColorPreprocessing','gray2rgb');
33
34 augimdsTrainof = augmentedImageDatastore(inputSize(1:2),
    imdsTrainof,'ColorPreprocessing','gray2rgb');
35
36 augimdsValidationA = augmentedImageDatastore(inputSize(1:2),
    imdsValidationA,'ColorPreprocessing','gray2rgb');
37 augimdsValidationE = augmentedImageDatastore(inputSize(1:2),
    imdsValidationE,'ColorPreprocessing','gray2rgb');
38 augimdsValidationD = augmentedImageDatastore(inputSize(1:2),
    imdsValidationD,'ColorPreprocessing','gray2rgb');
39 augimdsValidationEf = augmentedImageDatastore(inputSize(1:2),
    imdsValidationEf,'ColorPreprocessing','gray2rgb');
40 augimdsValidationDf = augmentedImageDatastore(inputSize(1:2),
    imdsValidationDf,'ColorPreprocessing','gray2rgb');
41
42 %Setando as Opcoes de treinamento
43 options = trainingOptions('sgdm', ...
44     'MiniBatchSize',10, ...
45     'MaxEpochs',6, ...
46     'InitialLearnRate',1e-4, ...
47     'ValidationData',augimdsTrainof, ...
48     'ValidationFrequency',3, ...
49     'ValidationPatience',Inf, ...
50     'Verbose',false, ...
51     'Plots','training-progress');
52
53
```

```
54 %Treinamento
55 netTransfer = trainNetwork(augimdsTrain,layers,options);
56
57
58 %Acurácias
59 [YPredA,scoresA] = classify(netTransfer,augimdsValidationA);
60
61 YValidationA = imdsValidationA.Labels;
62 accuracyA = mean(YPredA == YValidationA)
63
64 [YPredE,scoresE] = classify(netTransfer,augimdsValidationE);
65
66 YValidationE = imdsValidationE.Labels;
67 accuracyE = mean(YPredE == YValidationE)
68
69 [YPredD,scoresD] = classify(netTransfer,augimdsValidationD);
70
71 YValidationD = imdsValidationD.Labels;
72 accuracyD = mean(YPredD == YValidationD)
73
74 [YPredEf,scoresEf] = classify(netTransfer,augimdsValidationEf);
75
76 YValidationEf = imdsValidationEf.Labels;
77 accuracyEf = mean(YPredEf == YValidationEf)
78
79 [YPredDf,scoresDf] = classify(netTransfer,augimdsValidationDf);
80
81 YValidationDf = imdsValidationD.Labels;
82 accuracyDf = mean(YPredDf == YValidationDf)
83 end
```

# APÊNDICE C – ALGORITMO DE *TRANSFER LEARNING* DA GOOGLENET

```
1  function GoogleNet
2
3
4  %Carregando as imagens
5  imdsTrain = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\
       Base\Olhos\3Canais70\Dir','IncludeSubfolders',true,'
       LabelSource','foldernames');
6  imdsTrainof = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\
       Base\Olhos\3Canais15t\Dir','IncludeSubfolders',true,'
       LabelSource','foldernames');
7
8  imdsValidationA = imageDatastore('C:\Users\Convidado\Desktop\
       Lahiri\Base\Olhos\3Canais15v\Ambos','IncludeSubfolders',true,
       'LabelSource','foldernames');
9  imdsValidationE = imageDatastore('C:\Users\Convidado\Desktop\
       Lahiri\Base\Olhos\3Canais15v\Esq','IncludeSubfolders',true,'
       LabelSource','foldernames');
10 imdsValidationD = imageDatastore('C:\Users\Convidado\Desktop\
       Lahiri\Base\Olhos\3Canais15v\Dir','IncludeSubfolders',true,'
       LabelSource','foldernames');
11 imdsValidationEf = imageDatastore('C:\Users\Convidado\Desktop\
       Lahiri\Base\Olhos\3Canais15v\Esqflip','IncludeSubfolders',
       true,'LabelSource','foldernames');
12 imdsValidationDf = imageDatastore('C:\Users\Convidado\Desktop\
       Lahiri\Base\Olhos\3Canais15v\Dirflip','IncludeSubfolders',
       true,'LabelSource','foldernames');
13
14 %Carregando a GoogLeNet
15 net = googlenet;
16
17 %Parametros para a rede
18 inputSize = net.Layers(1).InputSize;
19 numClasses = numel(categories(imdsTrain.Labels));
20
21 %Adaptacao da GoogleNet para o problema escolhido
22 lgraph = layerGraph(net);
```

```
lgraph = removeLayers(lgraph, {'loss3-classifier','prob','output'});
newLayers = [
    fullyConnectedLayer(numClasses,'Name','fc','WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
    softmaxLayer('Name','softmax')
    classificationLayer('Name','classoutput')];
lgraph = addLayers(lgraph,newLayers);
lgraph = connectLayers(lgraph,'pool5-drop_7x7_s1','fc');

%Redimensionamento das imagens
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain,'ColorPreprocessing','gray2rgb');
augimdsTrainof = augmentedImageDatastore(inputSize(1:2),imdsTrainof,'ColorPreprocessing','gray2rgb');

augimdsValidationA = augmentedImageDatastore(inputSize(1:2),imdsValidationA,'ColorPreprocessing','gray2rgb');
augimdsValidationE = augmentedImageDatastore(inputSize(1:2),imdsValidationE,'ColorPreprocessing','gray2rgb');
augimdsValidationD = augmentedImageDatastore(inputSize(1:2),imdsValidationD,'ColorPreprocessing','gray2rgb');
augimdsValidationEf = augmentedImageDatastore(inputSize(1:2),imdsValidationEf,'ColorPreprocessing','gray2rgb');
augimdsValidationDf = augmentedImageDatastore(inputSize(1:2),imdsValidationDf,'ColorPreprocessing','gray2rgb');

%Setando as Opcoes de treinamento
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'ValidationData',augimdsTrainof, ...
    'ValidationFrequency',3, ...
    'ValidationPatience',Inf, ...
    'Verbose',false, ...
    'Plots','training-progress');

%Treinamento
```

```
53 netTransfer = trainNetwork(augimdsTrain,lgraph,options);
54
55 %Acurcias
56 [YPredA,scoresA] = classify(netTransfer,augimdsValidationA);
57
58 YValidationA = imdsValidationA.Labels;
59 accuracyA = mean(YPredA == YValidationA)
60
61 [YPredE,scoresE] = classify(netTransfer,augimdsValidationE);
62
63 YValidationE = imdsValidationE.Labels;
64 accuracyE = mean(YPredE == YValidationE)
65
66 [YPredD,scoresD] = classify(netTransfer,augimdsValidationD);
67
68 YValidationD = imdsValidationD.Labels;
69 accuracyD = mean(YPredD == YValidationD)
70
71 [YPredEf,scoresEf] = classify(netTransfer,augimdsValidationEf);
72
73 YValidationEf = imdsValidationEf.Labels;
74 accuracyEf = mean(YPredEf == YValidationEf)
75
76 [YPredDf,scoresDf] = classify(netTransfer,augimdsValidationDf);
77
78 YValidationDf = imdsValidationD.Labels;
79 accuracyDf = mean(YPredDf == YValidationDf)
80
81 end
```

# APÊNDICE D – ALGORITMO DE *TRANSFER LEARNING* DA RESNET18

```
function ResNet18


%Carregando as imagens
imdsTrain = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\
    Base\Olhos\3Canais70\Dir','IncludeSubfolders',true,'
    LabelSource','foldernames');
imdsTrainof = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\
    Base\Olhos\3Canais15t\Dir','IncludeSubfolders',true,'
    LabelSource','foldernames');

imdsValidationA = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Ambos','IncludeSubfolders',true,
    'LabelSource','foldernames');
imdsValidationE = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Esq','IncludeSubfolders',true,'
    LabelSource','foldernames');
imdsValidationD = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Dir','IncludeSubfolders',true,'
    LabelSource','foldernames');
imdsValidationEf = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Esqflip','IncludeSubfolders',
    true,'LabelSource','foldernames');
imdsValidationDf = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Dirflip','IncludeSubfolders',
    true,'LabelSource','foldernames');

%Carregando a ResNet18
net = resnet18;

%Parametros para a rede
inputSize = net.Layers(1).InputSize;
numClasses = numel(categories(imdsTrain.Labels));

%Adaptacao da ResNet18 para o problema escolhido
lgraph = layerGraph(net);
```

```
23 lgraph = removeLayers(lgraph, {'fc1000','prob','
       ClassificationLayer_predictions'});
24 newLayers = [
25     fullyConnectedLayer(numClasses,'Name','fc','
           WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
26     softmaxLayer('Name','smax')
27     classificationLayer('Name','classoutput')];
28 lgraph = addLayers(lgraph,newLayers);
29 lgraph = connectLayers(lgraph,'pool5','fc');
30
31 %Redimensionamento das imagens
32
33 augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain,
       'ColorPreprocessing','gray2rgb');
34 augimdsTrainof = augmentedImageDatastore(inputSize(1:2),
       imdsTrainof,'ColorPreprocessing','gray2rgb');
35
36 augimdsValidationA = augmentedImageDatastore(inputSize(1:2),
       imdsValidationA,'ColorPreprocessing','gray2rgb');
37 augimdsValidationE = augmentedImageDatastore(inputSize(1:2),
       imdsValidationE,'ColorPreprocessing','gray2rgb');
38 augimdsValidationD = augmentedImageDatastore(inputSize(1:2),
       imdsValidationD,'ColorPreprocessing','gray2rgb');
39 augimdsValidationEf = augmentedImageDatastore(inputSize(1:2),
       imdsValidationEf,'ColorPreprocessing','gray2rgb');
40 augimdsValidationDf = augmentedImageDatastore(inputSize(1:2),
       imdsValidationDf,'ColorPreprocessing','gray2rgb');
41
42 %Setando as Opcoes de treinamento
43 options = trainingOptions('sgdm', ...
44     'MiniBatchSize',10, ...
45     'MaxEpochs',6, ...
46     'InitialLearnRate',1e-4, ...
47     'ValidationData',augimdsTrainof, ...
48     'ValidationFrequency',3, ...
49     'ValidationPatience',Inf, ...
50     'Verbose',false, ...
51     'Plots','training-progress');
52
```

```
53 %Treinamento
54 netTransfer = trainNetwork(augimdsTrain,lgraph,options);
55
56 %Acurcias
57 [YPredA,scoresA] = classify(netTransfer,augimdsValidationA);
58
59 YValidationA = imdsValidationA.Labels;
60 accuracyA = mean(YPredA == YValidationA)
61
62 [YPredE,scoresE] = classify(netTransfer,augimdsValidationE);
63
64 YValidationE = imdsValidationE.Labels;
65 accuracyE = mean(YPredE == YValidationE)
66
67 [YPredD,scoresD] = classify(netTransfer,augimdsValidationD);
68
69 YValidationD = imdsValidationD.Labels;
70 accuracyD = mean(YPredD == YValidationD)
71
72 [YPredEf,scoresEf] = classify(netTransfer,augimdsValidationEf);
73
74 YValidationEf = imdsValidationEf.Labels;
75 accuracyEf = mean(YPredEf == YValidationEf)
76
77 [YPredDf,scoresDf] = classify(netTransfer,augimdsValidationDf);
78
79 YValidationDf = imdsValidationD.Labels;
80 accuracyDf = mean(YPredDf == YValidationDf)
81
82 end
```

# APÊNDICE E – ALGORITMO DE *TRANSFER LEARNING* DA RESNET50

```
function ResNet50


%Carregando as imagens
imdsTrain = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\
    Base\Olhos\3Canais70\Dir','IncludeSubfolders',true,'
    LabelSource','foldernames');
imdsTrainof = imageDatastore('C:\Users\Convidado\Desktop\Lahiri\
    Base\Olhos\3Canais15t\Dir','IncludeSubfolders',true,'
    LabelSource','foldernames');

imdsValidationA = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Ambos','IncludeSubfolders',true,
    'LabelSource','foldernames');
imdsValidationE = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Esq','IncludeSubfolders',true,'
    LabelSource','foldernames');
imdsValidationD = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Dir','IncludeSubfolders',true,'
    LabelSource','foldernames');
imdsValidationEf = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Esqflip','IncludeSubfolders',
    true,'LabelSource','foldernames');
imdsValidationDf = imageDatastore('C:\Users\Convidado\Desktop\
    Lahiri\Base\Olhos\3Canais15v\Dirflip','IncludeSubfolders',
    true,'LabelSource','foldernames');

%Carregando a ResNet50
net = resnet50;

%Parametros para a rede
inputSize = net.Layers(1).InputSize;
numClasses = numel(categories(imdsTrain.Labels));

%Adaptacao da ResNet50 para o problema escolhido
lgraph = layerGraph(net);
```

```
23 lgraph = removeLayers(lgraph, {'fc1000','fc1000_softmax','
       ClassificationLayer_fc1000'});
24 newLayers = [
25     fullyConnectedLayer(numClasses,'Name','fc','
           WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
26     softmaxLayer('Name','smax')
27     classificationLayer('Name','classoutput')];
28 lgraph = addLayers(lgraph,newLayers);
29 lgraph = connectLayers(lgraph,'avg_pool','fc');
30
31 %Redimensionamento das imagens
32
33 augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain,
       'ColorPreprocessing','gray2rgb');
34 augimdsTrainof = augmentedImageDatastore(inputSize(1:2),
       imdsTrainof,'ColorPreprocessing','gray2rgb');
35
36 augimdsValidationA = augmentedImageDatastore(inputSize(1:2),
       imdsValidationA,'ColorPreprocessing','gray2rgb');
37 augimdsValidationE = augmentedImageDatastore(inputSize(1:2),
       imdsValidationE,'ColorPreprocessing','gray2rgb');
38 augimdsValidationD = augmentedImageDatastore(inputSize(1:2),
       imdsValidationD,'ColorPreprocessing','gray2rgb');
39 augimdsValidationEf = augmentedImageDatastore(inputSize(1:2),
       imdsValidationEf,'ColorPreprocessing','gray2rgb');
40 augimdsValidationDf = augmentedImageDatastore(inputSize(1:2),
       imdsValidationDf,'ColorPreprocessing','gray2rgb');
41
42 %Setando as Opcoes de treinamento
43 options = trainingOptions('sgdm', ...
44     'MiniBatchSize',10, ...
45     'MaxEpochs',6, ...
46     'InitialLearnRate',1e-4, ...
47     'ValidationData',augimdsTrainof, ...
48     'ValidationFrequency',3, ...
49     'ValidationPatience',Inf, ...
50     'Verbose',false, ...
51     'Plots','training-progress');
52
```

```
53 %Treinamento
54 netTransfer = trainNetwork(augimdsTrain,lgraph,options);
55
56 %Acurcias
57 [YPredA,scoresA] = classify(netTransfer,augimdsValidationA);
58
59 YValidationA = imdsValidationA.Labels;
60 accuracyA = mean(YPredA == YValidationA)
61
62 [YPredE,scoresE] = classify(netTransfer,augimdsValidationE);
63
64 YValidationE = imdsValidationE.Labels;
65 accuracyE = mean(YPredE == YValidationE)
66
67 [YPredD,scoresD] = classify(netTransfer,augimdsValidationD);
68
69 YValidationD = imdsValidationD.Labels;
70 accuracyD = mean(YPredD == YValidationD)
71
72 [YPredEf,scoresEf] = classify(netTransfer,augimdsValidationEf);
73
74 YValidationEf = imdsValidationEf.Labels;
75 accuracyEf = mean(YPredEf == YValidationEf)
76
77 [YPredDf,scoresDf] = classify(netTransfer,augimdsValidationDf);
78
79 YValidationDf = imdsValidationD.Labels;
80 accuracyDf = mean(YPredDf == YValidationDf)
81
82 end
```