UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RAPHAEL RUSCHEL DOS SANTOS

DETECÇÃO DE ATAQUES DE APRESENTAÇÃO FACIAL UTILIZANDO REDES NEURAIS CONVOLUCIONAIS E INFORMAÇÕES DE CONTEXTO

Porto Alegre
2019

RAPHAEL RUSCHEL DOS SANTOS

DETECÇÃO DE ATAQUES DE APRESENTAÇÃO FACIAL UTILIZANDO REDES NEURAIS CONVOLUCIONAIS E INFORMAÇÕES DE CONTEXTO

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Engenharia da Computação

ORIENTADOR: Prof. Dr. Jacob Scharcanski

Porto Alegre
2019

RAPHAEL RUSCHEL DOS SANTOS

DETECÇÃO DE ATAQUES DE APRESENTAÇÃO FACIAL UTILIZANDO REDES NEURAIS CONVOLUCIONAIS E INFORMAÇÕES DE CONTEXTO

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Jacob Scharcanski, UFRGS
Doutor pela University of Waterloo – Waterloo, Canadá

Banca Examinadora:

Prof. Dr. Valner João Brusamarello, UFRGS
Doutor pela Universidade Federal de Santa Catarina

Prof. Dr. Letícia Viera Guimarães, UFRGS
Doutora pelo Muroran Institude of Technology – Muroran, Japão

Prof. Dr. Cristiano Bonato Both,
Doutor pela Universidade Federal do Rio Grande do Sul

Coordenador do PPGEE: _____
Prof. Dr. João Manoel Gomes da Silva Jr.

Porto Alegre, 26 de Abril de 2019.

AGRADECIMENTOS

Agradeço ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul (PPGEE - UFRGS) pela oportunidade de cursar o Mestrado acadêmico. Agradeço aos professores e funcionários da universidade, em especial meu orientador, Dr. Jacob Scharcanski por todo apoio e orientação dada durante a duração do programa. Além disso, agradeço meu colega de laboratório, Lucas Royes Schardosim, pelo apoio e por toda ajuda dada durante o desenvolvimento deste trabalho.

Gostaria de agradecer a minha família e amigos por todo o apoio dado durante estes dois anos e pela compreensão nos momentos de ausência.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

RESUMO

Sistemas de autenticação baseados em biometria são fortes candidatos a substituírem métodos de acesso tradicionais que utilizam nome de usuário e senha. Reconhecimento facial tem se tornado bastante popular nos últimos anos, e muitos dispositivos existentes já contam com uma câmera embutida, colaborando para sua popularização e tornando esta tecnologia fácil de ser utilizada. No entanto, sistemas de reconhecimento facial são suscetíveis a falhas de segurança, como os ataques de apresentação facial, onde um impostor tenta ganhar acesso ao sistema se disfarçando como um usuário genuíno, apesar de simples, estes ataques costumam ser bem-sucedidos. Na grande maioria dos trabalhos presentes na literatura, os autores utilizam apenas características faciais para realizar a detecção destes ataques. No entanto, este trabalho propõe que o quadro inteiro seja utilizado para o treinamento das redes neurais, visto que as informações dos arredores, como presença de dedos e molduras podem auxiliar a rede a discriminar estre um acesso genuíno e um impostor. Os resultados são positivos, utilizando a arquitetura GoogLeNet como base e usando o quadro inteiro, foi obtido um ganho de 4% em relação ao uso apenas das faces nos conjuntos de dados CASIA e NUAA, resultando em uma taxa de classificação correta média de 99%, superando os métodos da literatura que foram usados como referência.

Palavras-chave: Ataques de apresentação facial, Redes neurais, Aprendizado de máquina, Transferência de aprendizado.

ABSTRACT

Biometry-based authentication systems are potential candidates to replace traditional username and password-based access schemes. Facial recognition is becoming widely popular, and many existing devices already include embedded cameras, collaborating to its popularization and making this technology easy to use. Nevertheless, facial recognition systems are prone to security breaches, such as facial spoofing attacks, where an impostor tries to gain access to the system by disguising him/herself as a genuine user, despite being simple, these attacks tends to succeed. Most authors uses only the face to detect facial spoofing attacks. However, we present a novel way to train the neural network where the entire frame is used as input, arguing that the environment can contain rich information, such as fingers and frames, that are helpful to discriminate between a genuine and imposter frame. The preliminary experimental results are encouraging, and based on a GoogLeNet architecture and using the whole frame as input, we achieved an improvement of 4% on the correct classification rate over using only the faces for the CASIA and NUAA datasets, achieving an average of 99%, beating the methods used as reference.

Keywords: facial spoofing; convolutional neural networks; transfer learning; deep learning; presentation attack detection.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

COCO	Commom Objects in Context
CMYK	Cyan-Magenta-Yellow Key
CNN	Convolutional Neural Network
CCR	Correct Classification Rate
DET	Detection Error Tradeoff
FAR	False Acceptance Rate
FRR	False Rejection Rate
FC	Fully-Connected
GPU	Graphical Processing Unit
HTER	Half-Term Error Rate
ILSCRV	ImageNet Large-Scale Visual Recognition Challenge
CIE-LAB	Commission internationale de l'éclairage - Luminance, a-b
ML	Machine Learning
OCR	Optical Character Recognition
PAD	Presentation Attack Detection
RL	Regressão Linear
RNA	Redes Neurais Artificiais
ReLU	Rectified Linear Unit
SVM	Support Vector Machine
ROC	Receiver Operating Characteristic

SUMÁRIO

1 INTRODUÇÃO

Sistemas tradicionais de controle de acesso incluem métodos que utilizam *tokens*, como cartões de identificações, e esquemas baseados em conhecimento, como PINs e senhas. Apesar de populares, estas técnicas são suscetíveis a falhas no caso de uma pessoa mal-intencionada venha a tomar posse de tais credenciais, visto que neste caso é fácil realizar autenticação como se fosse um usuário genuíno do sistema. Como alternativa, tornou-se comum o uso de biometria para realizar controle de acesso. Esta abordagem consiste em utilizar alguma característica física ou comportamental de uma pessoa como forma de identificar um indivíduo visto que tais características biológicas costumam ser únicas em cada sujeito. Os sistemas biométricos podem basear seu funcionamento em diversas características, as mais comuns sendo, reconhecimento facial, de impressões digitais e de íris (JAIN; ROSS; PRABHAKAR, 2004).

Dentre os diversos métodos existentes, o reconhecimento facial tem se destacado dos demais por ser uma característica de captura fácil e não invasiva (SOLDERA et al., 2017). Além disto, a popularização de dispositivos como celulares, *tablet* e *laptops* com câmeras embutidas contribuíram para a proliferação desta técnica. Devido à popularização do reconhecimento facial, a comunidade científica começou a pesquisar sobre possíveis brechas de segurança presentes em sistemas deste tipo (DUC; MINH, 2009). Uma técnica para burlar estes procedimentos é conhecida como ataques de apresentação facial, ou ataques de *spoofing* facial. Este método consiste em apresentar ao sistema uma imagem do usuário genuíno, de forma a enganar o algoritmo e realizar autenticação como se fosse o usuário genuíno. Apesar de simples, esta abordagem costuma ter bastante sucesso.

Nesta dissertação, é avaliada uma nova maneira de realizar o treinamento de redes convolucionais para a tarefa de detecção de ataques de apresentação, onde o quadro inteiro é usado como entrada para a rede neural. Em geral, os métodos presentes na literatura utilizam apenas informações faciais para detectarem ataques de apresentação facial. Porém, quando a face é extraída do quadro, uma grande quantidade de informação sobre o ambiente é perdida, sendo que essas informações podem gerar pistas de contexto que são importantes para a identificação de ataques de apresentação, tanto que existem trabalhos como (KOMULAINEN; HADID; PIETIKAINEN, 2013) que utilizam tais informações. Para um humano, esta tarefa é feita de forma trivial quando o quadro inteiro é apresentado, no entanto, se apenas a face for apresentada, o problema se torna bem mais difícil. No entanto, na área de redes neurais, não foram encontrados trabalhos relacionados que façam o uso de pistas do ambiente para inferir o contexto da imagem. Para avaliar esta proposta, foram desenvolvidos modelos de redes neurais convolucionais (CNN, do inglês *Convolutional Neural Network*) construídos a partir de arquiteturas existentes fazendo o uso de técnicas de transferência de aprendizado, e seus desempenhos foram comparados, de forma a verificar se de fato a informação do ambiente é importante para a classifica-

ção. A utilização de CNNs é inspirada pelos resultados atingidos por essa abordagem em outras aplicações, sendo consideradas estado-da-arte em diversas tarefas de processamento de imagens e visão computacional (REDMON et al., 2016; HELD; THRUN; SAVARESE, 2016; LONG; SHELHAMER; DARRELL, 2015). Já o uso de transferência de aprendizado é justificado pela menor complexidade do projeto, visto que não é necessário modelar toda a arquitetura da rede. Dentre os benefícios desta metodologia destaca-se principalmente a possibilidade de realizar experimentos mais rapidamente, visto que não há necessidade de experimentar diversos parâmetros de arquitetura. Além disso, os modelos pré-treinados foram treinados com o objetivo de detectar milhares de classes e foram treinados com milhões de imagens, portanto, já aprenderem a extrair uma vasta quantidade de característica que podem ser úteis para a detecção de ataques de apresentação.

De maneira a comparar se este nova metodologia de treinamento é vantajosa em relação à abordagem comumente utilizada na literatura, foram realizados experimentos com CNNs treinadas utilizando imagens retiradas das bases de dados CASIA e NUAA, e também utilizando apenas as faces extraídas dos mesmos conjuntos de dados. Os experimentos realizados confirmam que é vantajoso a utilização do quadro inteiro, visto que os mapas de ativações apresentam valores elevados em áreas do ambiente, como dedos, braços, pescoço, roupas e molduras, mostrando que a rede de fato utiliza tais informações para gerar sua predição. Além disso, foi obtido um ganho na taxa de classificação correta de 4% em relação à utilização apenas das faces, tornando o modelo proposto competitivo com o estado-da-arte. Em conjunto ao ganho de desempenho em termos de métricas, também ocorre uma diminuição na complexidade total do algoritmo visto que as etapas de localização e extração da face não são mais necessárias.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, serão apresentadas as teorias das principais ferramentas utilizadas para o desenvolvimento desta dissertação. Inicialmente, serão introduzidos conceitos de ataques de apresentação facial e seus tipos. Em seguida, será realizada uma revisão de redes neurais convolucionais, posteriormente serão apresentados os modelos de CNN que foram utilizados como base para os experimentos. Por fim, serão apresentados conceitos de transferência de aprendizado de redes neurais.

2.1 Ataques de apresentação facial

Devido a popularização de métodos de identificação baseados em biometria, em especial, reconhecimento facial, esta abordagem se tornou o principal alvo de ataques de identificação. Estes também são conhecidos como ataques de apresentação, visto que o impostor se apresenta para um sistema como um usuário genuíno ao tentar reproduzir suas credenciais utilizando diversas táticas para enganar o sistema.

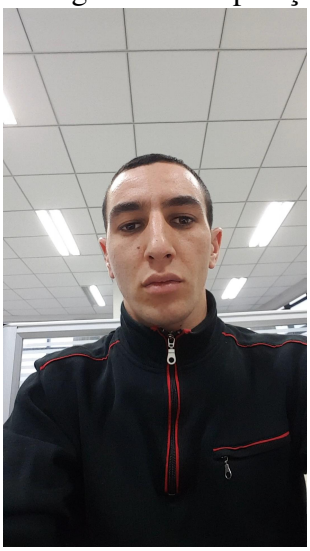
Detecção de *spoofing* facial, detecção de vivacidade facial, contra medidas para ataques de *spoofing* facial e detecção de ataques de apresentação (PAD, do inglês *Presentation Attack Detection*) são termos usados de forma intercambiável para descrever o problema de detectar quando um impostor está tentando se passar por um usuário genuíno de um sistema de reconhecimento facial (SOUZA et al., 2018). Este é um tipo de ataque que ocorre a nível de sensor, e não por meio de *software*, como ataques de força bruta para descobrir a senha de um usuário. Devido a sua complexidade, muitas vezes é difícil até mesmo para um humano distinguir entre uma face legítima e uma falsa. Uma comparação lado-a-lado entre uma imagem genuína e um ataque de foto impressa é mostrada na Figura 1. Dentre os diversos tipos de ataques que podem ser realizados por impostores, os mais estudados são os seguintes tipos (GALBALLY; MARCEL; FIERREZ, 2014a):

- **Ataques de foto impressa**: Este tipo de ataque é o mais comum e consiste na utilização de uma foto impressa contendo o rosto do usuário genuíno do sistema. A popularidade deste método se deve principalmente a facilidade de se encontrar uma foto facial de um certo sujeito utilizando a internet, principalmente em redes sociais.

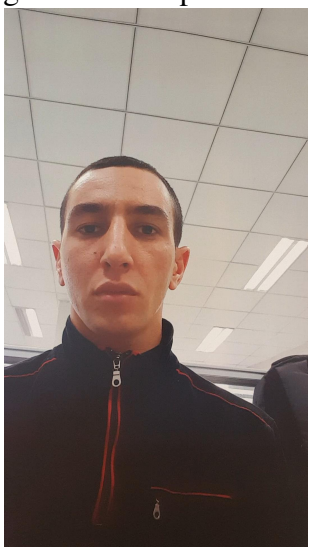
- **Fotos com recortes nos olhos**: Este tipo é similar ao ataque de foto impressa, porém, os olhos são recortados de forma que o impostar possa imitar o padrão de piscadas do usuário real.

- **Foto deformada**: Do inglês *warped photo*, consiste em um tipo de ataque similar aos anterior, porém, o impostar movimenta a foto impressa de forma a simular

Figura 1: Comparação entre um quadro original e um impostor.



(a) Imagem genuína;



(b) Imagem de foto impressa.

Fonte: (ZHANG et al., 2012).

movimentos faciais.

- **Apresentação de vídeo**: Este tipo de ataque é feito mostrando um vídeo contendo o usuário genuíno ao sistema de reconhecimento facial por meio de um *tablet* ou *smartphone*. Esta categoria apresenta quase todos os comportamentos similares à uma face real, como movimento facial, piscadas, etc.

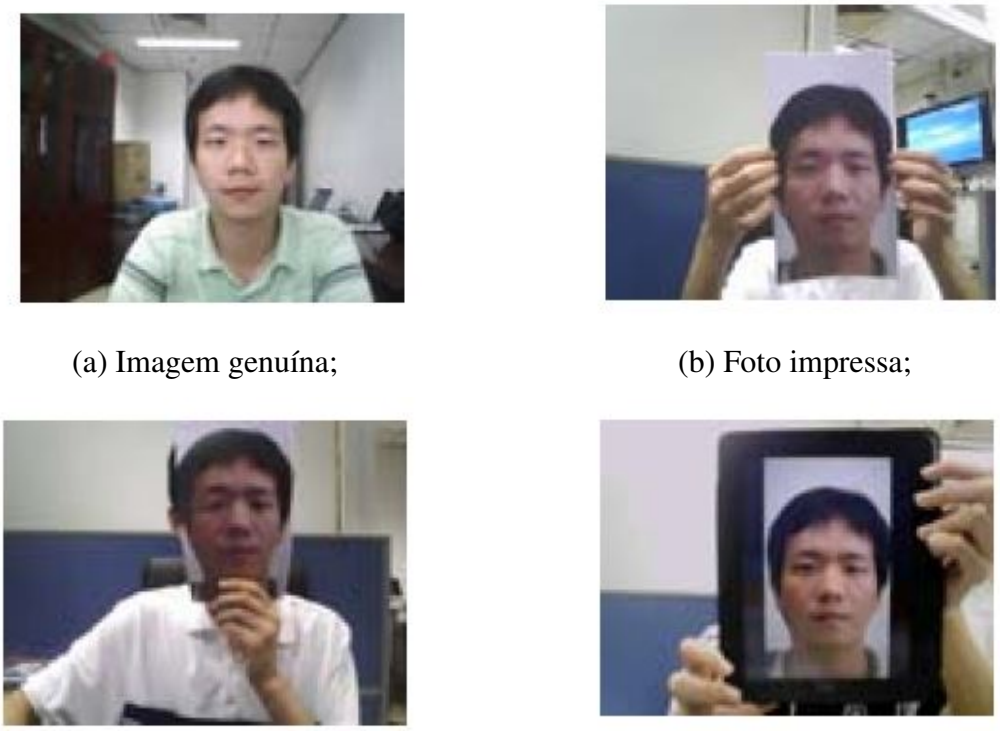
- **Máscaras**: Este tipo de ataque é geralmente utilizando quando o sistema de detecção de *spoofing* que analisam estruturas faciais 3D. Este tipo de ataque também apresenta todas as características de uma face real e portanto é o mais difícil de ser detectado. No entanto, é o ataque mais difícil de ser realizado, pois necessita a fabricação da máscara semelhante ao usuário genuíno. Ataques de máscaras não foram considerados durante este trabalho visto que os *datasets* utilizados não contém dados deste tipo de ataque, além de geralmente ser necessário a utilização de *hardware* extra para a detecção deste ataque.

Um exemplo de imagens de um usuário genuíno é apresentado na Figura 2 (a), e os três primeiros tipos de ataques são mostrados na Figura 2 (b)-(d), onde é possível notar que quando um ataque de apresentação está sendo realizado, a área ao redor da face contém informações que podem ser utilizadas para inferir se um dado quadro é de fato uma tentativa de *spoofing*. Tal informação inclui a presença de dedos ao redor da face, presença de bordas de dispositivos móveis, artefatos decorrentes do processo de reamostragem devido a impressão ou apresentação de vídeo, que não estão presentes em outras áreas da imagem, além de diferentes propriedades de refletância presentes na face, mas que não persistem no resto do corpo.

2.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais, ou CNNs, são inspiradas em modelos biológicos do cérebro, assim como as RNAs. Durante a década de 1950 e 1960, pesquisadores

Figura 2: Exemplos de um quadro genuíno e de 3 tipos de ataques.



(a) Imagem genuína; (b) Foto impressa;

(c) Foto com recorte nos olhos; (d) Apresentação de vídeo.

Fonte: (ZHANG et al., 2012)

sugeriram um novo modelo para explicar como mamíferos percebem o mundo a partir de sua visão. Dentre suas descobertas, eles mostraram que o córtex visual tanto de gatos quanto de macacos possuem neurônios que respondem ao ambiente.

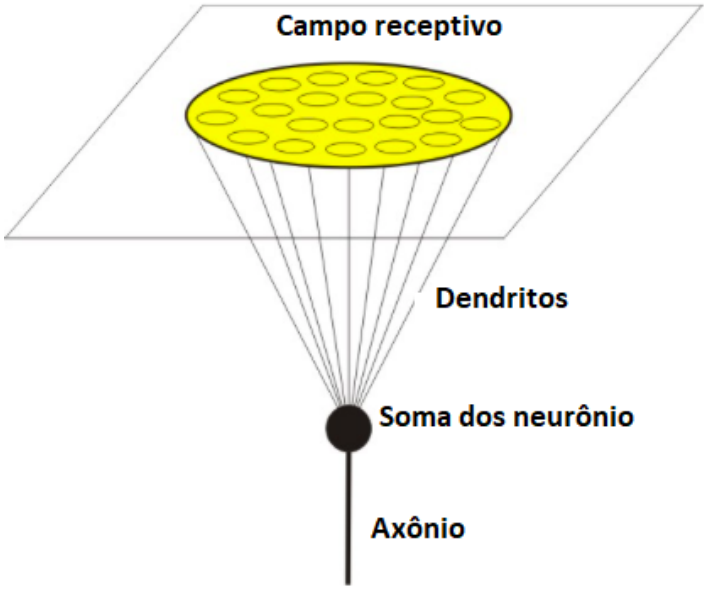
Em (HUBEL; WIESEL, 1968), os autores descreveram dois tipos básicos de células visuais no cérebro e que agem de forma diferente: células simples (S) e células complexas (C). As células do tipo S ativam quando identificam formatos básicos, como linhas em uma área e ângulo específico. Já as do tipo C contém um maior campo receptivo e são capazes de ativar independentemente da posição e do formato do objeto. No sentido da visão, um campo receptivo de um único neurônio sensorial é uma região específica da retina que afeta a ativação desse neurônio. Todas as células sensoriais possuem campos receptivos semelhantes que se sobrepõem. Uma ilustração deste processo é mostrado na Figura 3.

Baseados nessas ideias, em 1999, as CNNs foram introduzidas em um artigo por (LECUN et al., 1999), onde uma rede chamada de LeNet-5 era capaz de classificar números escritos à mão.

2.2.1 Motivação

As CNNs possuem uma grande semelhança com as redes *Multilayer Perceptron*. Ambas são compostas de neurônios com pesos de conexão que são ajustados durante o processo de treinamento, sendo que cada neurônio realiza uma soma ponderada das entradas com seus respectivos pesos e em seguida utilizam uma função de ativação. A grande diferença é que as arquiteturas de CNNs assumem que as entradas sempre são imagem,

Figura 3: O campo receptivo de um neurônio.



Fonte: Medium [1]

e isso permite que certas propriedades sejam inseridas na arquitetura de forma a tornar a rede mais eficiente. Nas RNAs tradicionais, as entradas são vetores de características, e estes vetores são transformados por uma série de neurônios em camadas escondidas, aonde cada nó é totalmente conectado a todos os nós da camada anterior. Esta arquitetura onde cada nó é totalmente conectado com todos os neurônios da camada anterior faz com que este tipo de rede não seja recomendado para o uso em imagens.

Um exemplo deste problema pode ser visto utilizando como base o conjunto de dados CIFAR-10, criado pela *Canadian Institude For Advanced Research*, e cujos dados consistem em imagens de tamanho 32x32 *pixels* com 3 canais de cores. A partir do tamanho da imagem, cada neurônio na primeira camada da rede irá conter $32 \times 32 \times 3 = 3072$ pesos. Esta quantidade de pesos pode não ser um número tão expressivo considerando o poder computacional disponível atualmente. No entanto, imagens de tamanho 32x32 raramente são utilizadas atualmente. Para uma imagem mais moderna, de tamanho 200×200 e 3 canais de cores, haveria a necessidade de 120.000 parâmetros para cada neurônio na camada de entrada. Além disso, as RNAs são compostas de diversas camadas contendo diversos neurônios cada, resultando num número gigantesco de parâmetros que seriam necessários treinar, é possível concluir que além de ser um processo demorado, também seria suscetível a *overfitting*.

Além do problema de dimensionalidade, as RNAs não levam em conta a estrutura espacial dos dados de entrada, deste modo elas tratam *pixels* que estão longe da mesma maneira que *pixels* que estão perto. Para muitos tipos de dados, este detalhe pode não representar problemas, porém em imagens a estrutura espacial fornece informações de muita importância.

Para contornar este problema, as camadas convolucionais são organizadas em 3 dimensões: altura, largura e profundidade. Além disso, baseado na ideia dos campos receptivos, os neurônios em uma camada não se conectam a toda entrada, e sim a apenas uma

[1] Disponível em : <https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050>. Acesso em: 3/5/2019

região dela.

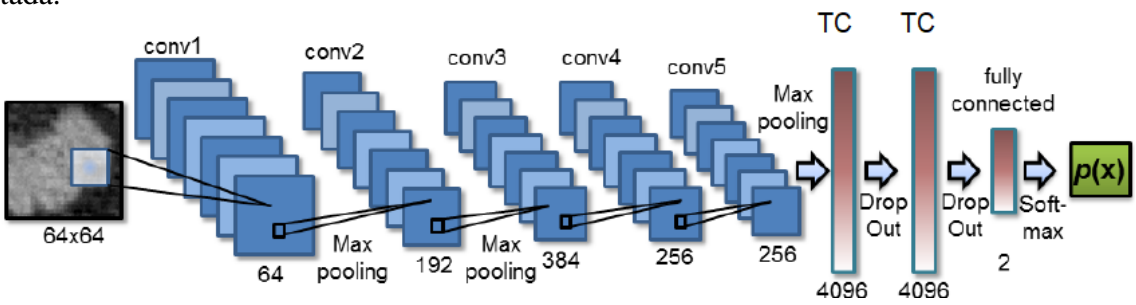
2.2.2 Arquitetura das CNNs

Assim como as RNAs, as CNNs também são formadas juntando diversos tipos de camadas. As principais camadas que compõem uma Rede Neural Convolucional são:

- Camada de entrada: Contém os valores do *pixels* da imagem de entrada, logo, seu tamanho será o mesmo dela, por exemplo 32x32x3 no caso do conjunto CIFAR-10.
- Camada convolucional: Esta camada é a responsável por calcular os valores de saídas de cada neurônio que está conectado à uma certa região do volume de entrada através da convolução com os filtros contidos nesta estrutura.
- Camadas de *pooling*: Irá realizar uma subamostragem nas dimensões espaciais do seu volume de entrada.
- Completamente conectadas: Também conhecidas como *Multilayer Perceptron*, são utilizadas para calcular o valor de saída da rede, em geral, a pontuação de cada classe.

Um exemplo de uma arquitetura de CNN contendo as camadas mostradas acima é apresentada em 4.

Figura 4: Ilustração de uma típica arquitetura de CNN. ”Conv” indica uma camada convolucional, ”Pool” indica uma camada de Pooling e ”TC” uma camada totalmente conectada.



Fonte: (ROTH et al., 2015)

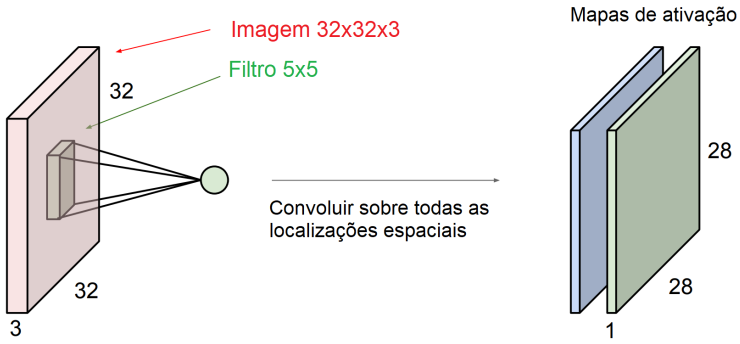
Devido à importância das camadas convolucionais e de *pooling*, elas serão apresentadas em mais detalhes a seguir.

2.2.2.1 Camadas convolucionais

As camadas convolucionais são a parte mais importante de uma CNN, visto que são nessas camadas que grande parte do processamento é de fato realizado. De maneira similar as camadas das RNAs, as camadas convolucionais também consistem em um conjunto de parâmetros que são ajustados durante o processo de treinamento da rede, porém, ao invés de serem constituídos por pesos de conexão entre neurônios, estas camadas consistem em um conjunto de filtros digitais. Os filtros tipicamente empregados nestas camadas possuem dimensão espacial reduzida (altura e largura), mas se estendem por toda a profundidade do volume de entrada. Por exemplo, um filtro aplicado na camada de entrada pode ser de tamanho 7x7x3, ou seja, 7 *pixels* de largura e de altura, e 3 de profundidade que é o mesmo número de canais presentes em uma imagem digital RGB.

Durante a fase onde as entradas são propagadas pela rede, cada volume de entrada é convoluído com o conjunto de filtros da camada convolucional seguinte. Conforme o filtro vai sendo passado pelo volume, o resultado será uma matriz de mapas de ativação que irá representar a resposta do filtro para uma dada posição espacial. Ao final do processo estes mapas de ativação são concatenados de forma a gerar o volume de saída para aquela camada, conforme ilustrado na Figura 5.

Figura 5: Exemplo do funcionamento de uma camada convolucional.



Fonte: Stanford [2]

Tipicamente, a rede irá aprender filtros que possuam altas ativações quando encontrarem um tipo de característica visual de baixo nível como bordas e cantos nas primeiras camadas e conforme aumenta a profundidade da rede, maior será o nível de abstração das feições.

Como visto na Subseção 2.2.1, quando os dados de entrada possuem dimensionalidade elevada, o uso das RNAs tradicionais se torna impraticável devido ao número elevado de parâmetros que devem ser ajustados de modo a ter uma rede totalmente conectada. No entanto, as CNNs se conectam à apenas uma pequena região de volume de entrada e o tamanho desta conexão é um hiperparâmetro conhecido como campo receptivo, nome inspirado pelo funcionamento de sistemas biológicos. Para ilustrar essa conectividade local, suponha uma imagem de entrada de tamanho 32x32 *pixels* e 3 canais correspondentes ao RGB. Se o campo receptivo (tamanho do filtro) for 5x5, então cada neurônio na camada convolucional irá ter conexões com uma região de tamanho 5x5x3 no volume de entrada, totalizando $5 * 5 * 3 = 75$ pesos. Nota-se que a profundidade do filtro deve ser 3 devido ao número de canais presentes na imagem de entrada, ou seja, a profundidade do filtro sempre possui a mesma dimensão da profundidade do volume de entrada, e portanto costuma ser omitida. A Figura 6 ilustra este processo de conectividade local.

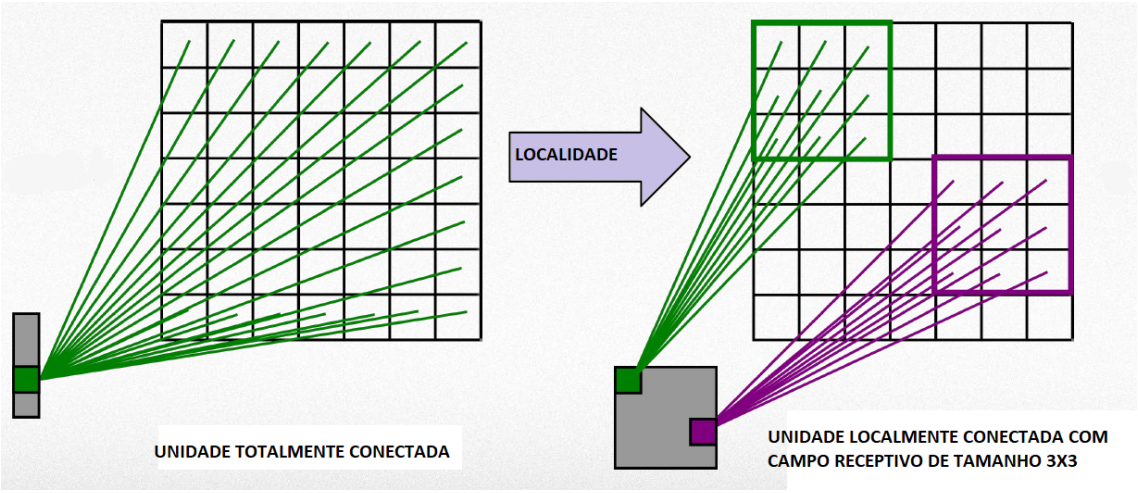
Além do tamanho do filtro, existem outros três hiperparâmetros que também devem ser selecionados e que controlam o tamanho do volume de saída de cada camada, são eles:

- **Profundidade**: Este parâmetro corresponde ao número de filtros que será usado em cada camada, cada filtro será responsável por procurar características diferentes no sinal de entrada. Por exemplo, supondo que a primeira camada convolucional possua 3 filtros, cada um destes pode procurar por bordas em diferentes orientações.

- **Passo**: Este valor indica a forma que a convolução será realizada. Se este parâmetro tiver valor igual a 1, então o filtro se move 1 *pixel* por vez. Se o passo for igual a 2,

[2]Disponível em : <http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf>. Acesso em: 3/5/2019

Figura 6: Ilustração do processo de conectividade local de uma CNN.



Fonte: Github.io [3]

então o filtro irá pular 2 pontos cada vez que for movido, e assim sucessivamente. Quanto maior o valor do passo, menor será o volume de saída.

- **Preenchimento com zeros**: Dependendo do tamanho do filtro e do tamanho do passo, poderá ocorrer a necessidade de avaliar o valor do filtro em uma região fora da imagem. Para que este processo possa ocorrer, deve-se preencher com zeros a região das bordas da imagem. O número de linhas e colunas que serão preenchidas são dados por este parâmetro.

A partir destes dados, é possível calcular o tamanho do volume de saída utilizando a seguinte equação:

$$T = \frac{W - F + 2P}{S + 1} \tag{1}$$

Onde W representa o tamanho do volume de entrada, F o tamanho do campo receptivo, P a quantidade de preenchimento com zeros e S o tamanho do passo. É importante ressaltar que nem todas as combinações destes fatores são possíveis, visto que o tamanho do volume de saída deve ser um número inteiro.

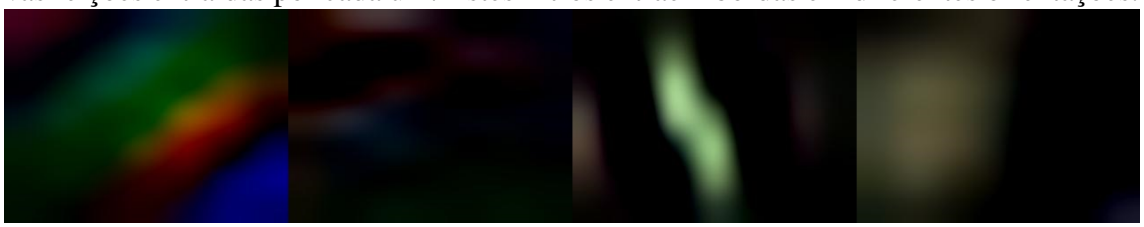
A partir do comportamento das camadas convolucionais, é possível concluir que elas trabalham como extratoras de características, e não como classificadores, e por isso há a necessidade de adicionar camadas totalmente conectadas ao final da arquitetura. Um exemplo dos filtros aprendidos por uma CNN e as feições extraídas por estes filtros são mostrados na Figura 7.

2.2.2.2 Camadas de pooling

Comumente utilizada em arquiteturas de redes convolucionais é a chamada camada de *pooling*. Sua principal função é reduzir as dimensões espaciais, menos a profundidade. O *pooling* funciona como uma forma de subamostragem não-linear, havendo diversos métodos de realizar este procedimento, como média, norma l^2, porém, o mais comum é o chamado *Max pooling*. A forma com que este processo funciona é: a entrada é

[3] Disponível em : <http://i-systems.github.io/HSE545/machinelearningall/Workshop/KIMM_2018_WINTER/07CNN.html>. Acesso em: 3/5/2019

Figura 7: Exemplo de filtros aprendidos pela primeira camada convolucional e as respectivas feições extraídas por cada um. Estes filtros extraem bordas em diferentes orientações.



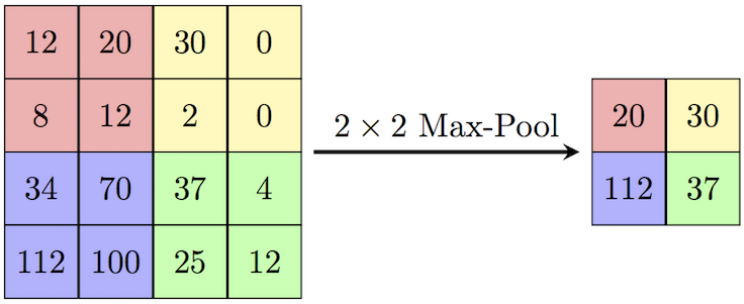
(a) Filtros 7×7;

(b) Feições extraídas;

Fonte: Autor

particionada em diferentes retângulos sem sobreposição e o maior valor em cada retângulo é transferido como resultado. Um exemplo deste processo é mostrado na Figura 8.

Figura 8: Exemplo de *Max pooling*.



Fonte: Computer Science Wiki [4]

Quando a posição do objeto na cena não é de interesse, o uso de *pooling* se torna interessante devido a redução na dimensionalidade das entradas. Essa diminuição tem como principais benefícios a redução do número de parâmetros a ser treinado, o que leva a uma diminuição no processamento necessário, redução de memória utilizada e diminui a chance de ocorrer *overfitting*. Além disso, ao adicionar este procedimento, também provê um certo grau de invariância a translação para a rede. Uma prática comum no projeto de CNNs é a utilização de uma camada de *pooling* após cada camada convolucional.

2.3 Modelos de redes neurais convolucionais

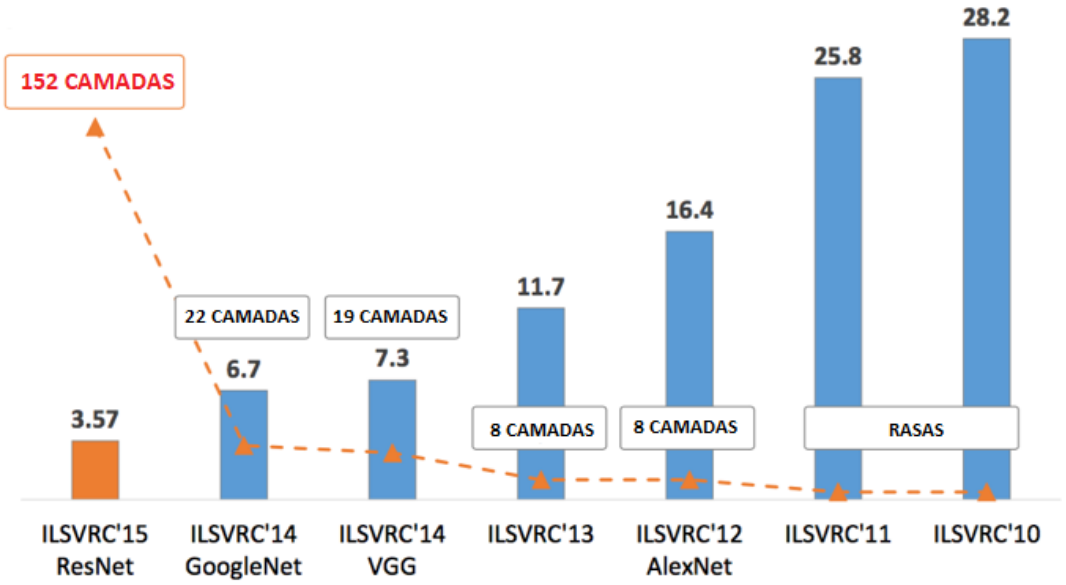
Visto que o problema de detecção de ataques faciais consiste em separar as categorias genuínas das impostoras. Com base na literatura disponível, as arquiteturas mais

[4]Disponível em : <https://computersciencewiki.org/index.php/Max-pooling_/_Pooling>. Acesso em: 3/5/2019

populares são LeNet-5 (LECUN et al., 1999), AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), GoogLeNet (SZEGEDY et al., 2015) e ResNet (HE et al., 2016).

Estas redes se tornaram populares devido a seu desempenho no *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*, que é um desafio que ocorre anualmente para que os participantes apresentem suas novas técnicas de reconhecimento de objetos. Apesar do *dataset* ImageNet conter dezenas de milhões de amostras separadas em 22.000 classes, durante o desafio do ILSVRC são utilizados apenas mil imagens para cada uma das mil classes selecionadas, totalizando 1.2 milhões de imagens, sendo 50 mil para validação, 150 mil para testes e o restante para treinamento. O desempenho de cada rede e seu respectivo número de camadas é apresentado na Figura 9.

Figura 9: Percentual de erro apresentado por cada rede no desafio ILSVRC e seu respectivo número de camadas.



Fonte: Medium [5]

A arquitetura AlexNet foi selecionada para ser testada devido a sua popularidade por ser a primeira CNN a obter taxa de erro inferior a 20% no desafio, além disso, devido à quantidade menor de camadas, seu treinamento é mais rápido que as demais. Como segunda arquitetura a ser testada, foi selecionada a GoogLeNet devido a um bom balanço entre número de camadas e taxa de erro. Quando comparada com a ResNet, a GoogLeNet possui uma taxa de erro 87% maior, porém, possui praticamente 7 vezes menos camadas, fazendo com que seu treinamento seja várias ordens de magnitude mais rápido que a ResNet. A rede VGG (LIU; DENG, 2015), em um primeiro momento pode parecer um bom candidato a ser testada mas apesar de possui apenas 19 camadas, o número de parâmetros que deve ser treinado é de 138 milhões, contra apenas 4 milhões da GoogLeNet.

A seguir são apresentadas as arquiteturas de redes convolucionais utilizadas nessa dissertação.

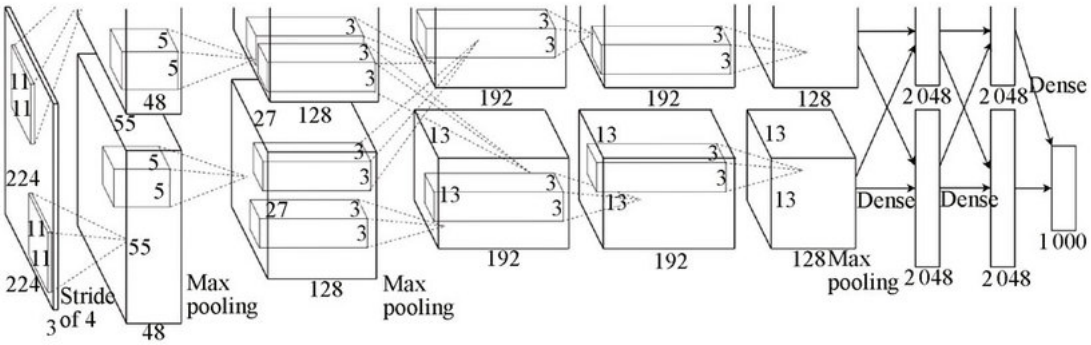
[5]Disponível em : <https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet-7c81c017b848>. Acesso em: 3/5/2019

2.3.1 AlexNet

Em 2012, a comunidade acadêmica da área de visão computacional, em especial de reconhecimento de objetos, ficou admirada quando a arquitetura de rede convolucional proposta por (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) diminui a taxa de erro do ILSVRC de 25.8% para 16.4%, tornando-se o primeiro método a obter uma taxa menor que 20%. Este momento é considerando um marco histórico para a área e é considerado como um dos pilares para a popularização das CNNs nos últimos anos.

A arquitetura AlexNet é composta por um total de 8 camadas, sendo 5 camadas convolucionais seguidas de camadas de *pooling* para extrair as feições e 3 camadas totalmente conectadas seguidas de uma camada *softmax* com 1000 neurônios para realizar a classificação entre as mil classes do desafio. Com este número de camadas, a rede possui 650.000 neurônios e 60 milhões de parâmetros a serem treinados. Uma ilustração da arquitetura da AlexNet é mostrada na Figura 10, onde *'dense'* se refere à camadas totalmente conectadas.

Figura 10: Arquitetura AlexNet.



Fonte: (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

De acordo com os autores, o treinamento desta rede foi feita com 1.2 milhões de imagens e o tempo necessário para o treinamento foi de 6 dias utilizando duas GPUs nVidia GTX 580 com 3GB de memória interna. O autor também nota que o tamanho da rede foi limitado pela disponibilidade de memória na GPU, e segundo seus testes, retirar uma camada convolucional acarreta num aumento de 2% na taxa de erro encontrada.

2.3.2 GoogLeNet

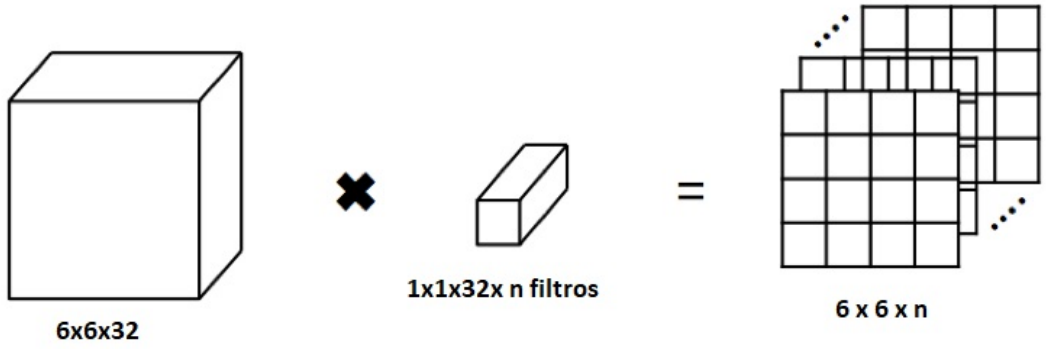
A arquitetura entitulada GoogLeNet, como o nome sugere, foi criada por um time de desenvolvedores do Google, e a terminação "LeNet"serve como um tributo a rede LeNet que foi criada por Yan LeCun e é considerada a pioneira da área de redes convolucionais. Esta arquitetura venceu o desafio ILSVRC de 2014, com uma grande margem sobre a ZFNet de (ZEILER; FERGUS, 2014) que foi a vencedora de 2013 e sobre a AlexNet. Após a vitória, detalhes da arquitetura foram publicados em (SZEGEDY et al., 2015).

As grandes diferenças desta arquitetura são a utilização de convoluções de tamanho 1×1 no meio da rede, utilização de uma técnica de *pooling* médio global ao final da rede, e utilização do chamado módulo *inception*.

As convoluções 1×1 são utilizadas para introduzir mais não-linearidades ao modelo, aumentando o poder de representatividade dele. Além disso, a GoogLeNet emprega este esquema como um modo de reduzir a dimensionalidade de forma a diminuir o custo computacional necessário. O resultado de uma convolução 1×1 é similar ao efeito de

pooling, porém aplicado à profundidade do volume de entrada, ou seja, ao invés de reduzir a resolução espacial do volume (altura e largura), a profundidade que é diminuída ao realizar a convolução com o filtro. Visto que o kernel sempre irá possuir tamanho $1 \times 1 \times d$, onde d representa a profundidade da camada anterior (nota-se que como citado na Seção 2.2, a profundidade do filtro costuma ser omitida pois ela é sempre igual a profundidade do volume o qual o filtro está sendo aplicado). Um exemplo deste processo pode ser visto na Figura 11.

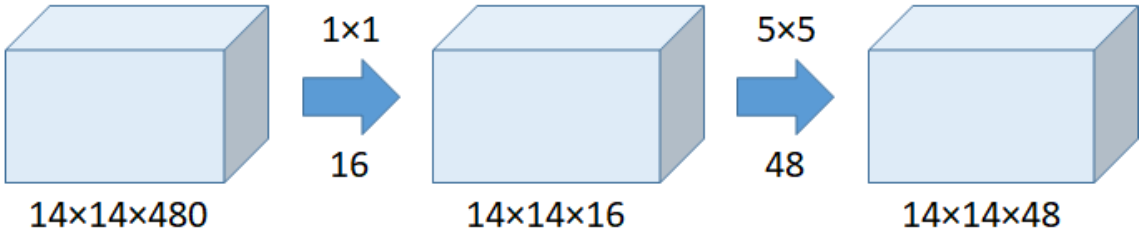
Figura 11: Exemplo do processo de redução de profundidade realizado pela convolução 1×1.



Fonte: Medium [6]

Esta redução na profundidade do volume acarreta numa diminuição do número de operações necessárias. Por exemplo, suponha um volume de entrada de tamanho $14 \times 14 \times 480$ no qual será aplicado 48 filtros 5×5, resultando em um volume de saída de dimensão $14 \times 14 \times 48$. O número de operações necessárias para realizar esta operação será $(14 \times 14 \times 48) \times (5 \times 5 \times 480) = 112.9M$. Adicionando uma camada convolucional intermediária com 16 filtros de tamanho 1×1 com o objetivo de reduzir a profundidade do volume de 480 para 16, como mostrado na Figura 12, o número de operações será:

Figura 12: Exemplo da adição de uma camada de convolução 1×1 intermediária.



Fonte: Medium [7]

- Para a camada 1×1: $(14 \times 14 \times 16) \times (1 \times 1 \times 480) = 1.5M$

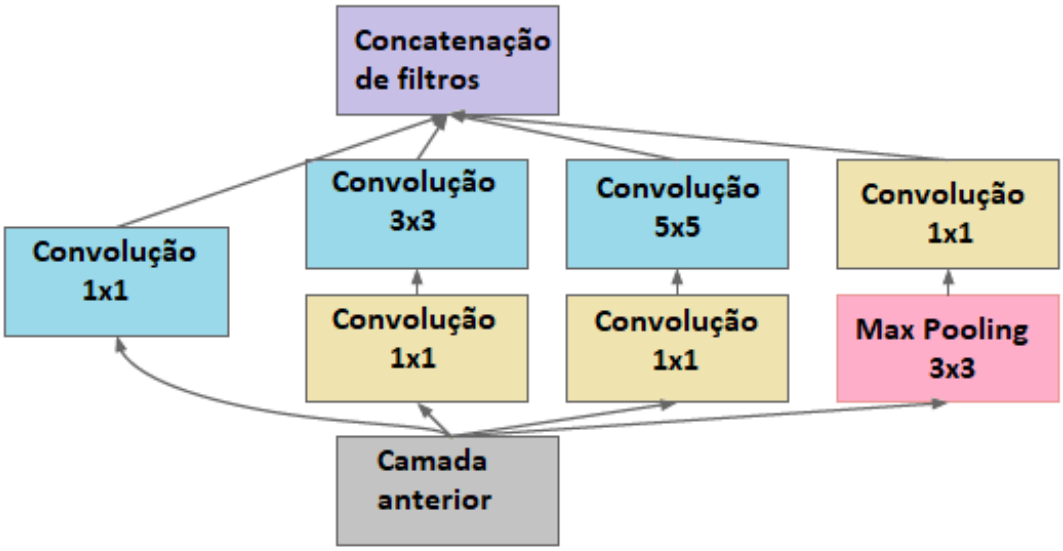
[6]Disponível em : <https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524>. Acesso em: 3/5/2019

- Para a camada 5×5: $(14 \times 14 \times 48) \times (5 \times 5 \times 16) = 3.8M$

O que resulta num total de $5.3M$ operações, praticamente 21 vezes menor que o número de operações necessárias anteriormente.

Nas arquiteturas tradicionais de CNNs, o tamanho do filtro de cada camada costuma ser um parâmetro a ser escolhido durante o projeto e utiliza-se apenas um caminho de convolução em cada camada, assim como na AlexNet. Devido a redução do número de cálculos necessários proveniente da utilização da convolução 1×1, foi possível inserir diversos caminhos em uma mesma camada, criando o chamado módulo *inception* que é ilustrado na Figura 13.

Figura 13: Ilustração do módulo *inception* utilizado na GoogLeNet.



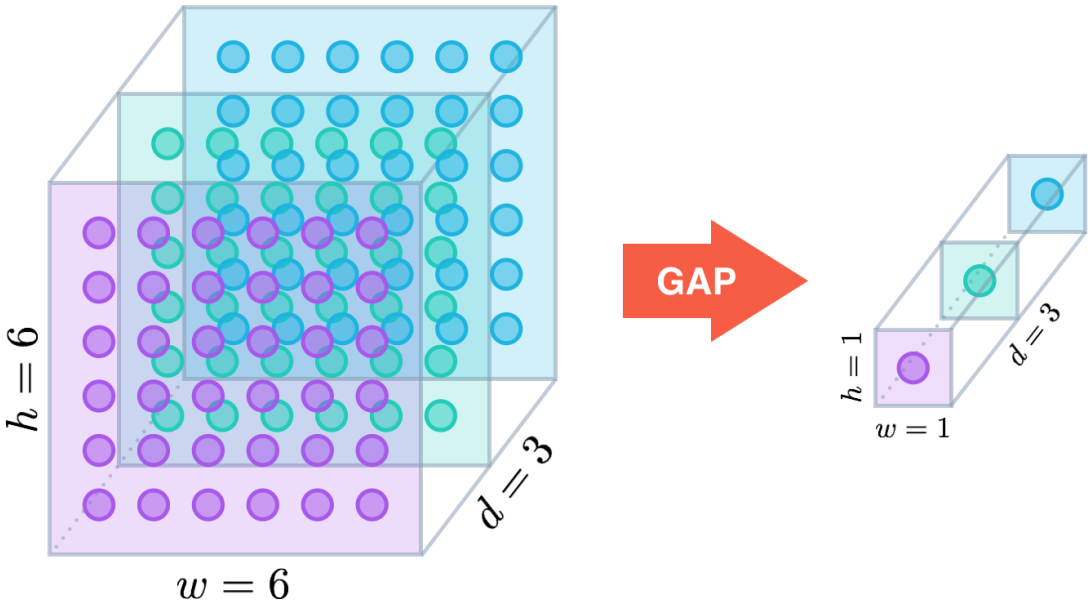
Fonte: (SZEGEDY et al., 2015)

A ideia principal desse módulo é ser possível explorar áreas maiores com os filtros 5×5, mas o mesmo tempo possuir uma boa resolução em áreas menores, utilizando os filtros 1×1 e 3×3. Para isso, são inseridos múltiplos caminhos com filtros diferentes para a mesma entrada e em seguida as feições extraídas por cada caminho são concatenadas. Além da vantagem de analisar a entrada em múltiplas escalas ao mesmo tempo, ao inserir diversos filtros em paralelo remove a necessidade de selecionar o parâmetro referente ao tamanho do filtro, deixando com que a rede aprenda qual o melhor caminho para solucionar o problema.

Para reduzir ainda mais o número de parâmetros e assim diminuir a chance de ocorrer *overfitting*, os engenheiros fizeram a utilização de *Pooling* médio global (do inglês, *Global Average Pooling*, ou GAP). Esta técnica se comporta de maneira similar à uma camada de *max pooling*, porém, a diferença se deve ao fato que a saída de uma camada GAP é a média dos valores contidos em cada canal, ou seja, para um tensor com dimensões $h \times w \times d$, a sua nova dimensão será $1 \times 1 \times d$. Uma ilustração desse processo é apresentado na Figura 14, onde cada um dos pontos coloridos na saída representa a média de todos os valores com a mesma cor no volume de entrada.

[7]Disponível em : <https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524>. Acesso em: 3/5/2019

Figura 14: Exemplo do processo de *Pooling* Médio Global.



Fonte: Github.io [8]

A arquitetura completa com as 22 camadas presentes na rede GoogLeNet é mostrada nas Figura A no Anexo A

2.4 Transferência de aprendizado

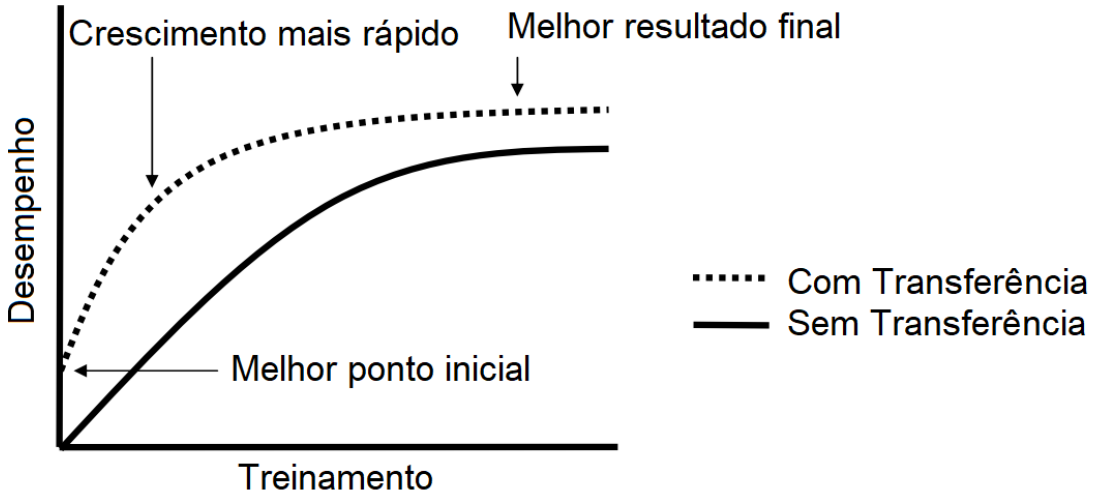
Para realizar o treinamento das CNNs, foi utilizado uma técnica conhecida como transferência de aprendizado (do inglês *Transfer Learning*) que é uma metodologia de aprendizado de máquina onde um modelo treinado para uma determinada aplicação é reutilizado para uma outra tarefa semelhante (PRATT, 1993). Por exemplo, uma rede que aprendeu a reconhecer carros, pode ser reutilizado para criar um novo modelo para reconhecer caminhões. É um método bastante popular em aprendizado profundo, em especial visão computacional e processamento de linguagem natural, onde ao invés de iniciar o treinamento da rede neural do zero, utiliza-se um outro modelo pré-treinado como ponto inicial com intuito de reduzir o custo computacional e o tempo necessário para completar o treinamento.

Ao utilizar esta metodologia de transferência, o conhecimento obtido durante o treinamento do modelo original é aproveitado para facilitar o aprendizado da nova tarefa. Com isso, costuma-se obter diversos benefícios, os principais sendo: menor tempo de treinamento, menor número de amostras necessárias, menor número de hiperparâmetros para serem ajustados, ponto inicial mais favorável, e em geral, resultado final melhor, conforme apresentado na Figura 15.

Em transferência de aprendizado, primeiramente uma rede base é treinada em um conjunto de dados base para uma certa tarefa, então as feições aprendidas são reutilizadas, ou transferidas, para que uma segunda rede seja treinada em um outro conjunto de dados para realizar uma nova tarefa. No caso das CNNs, costuma-se reutilizar apenas as cama-

[8] Disponível em : <https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/>. Acesso em: 3/5/2019

Figura 15: Benefícios da utilização de transferência de aprendizado.



Fonte: (AYTAR; ZISSERMAN, 2011)

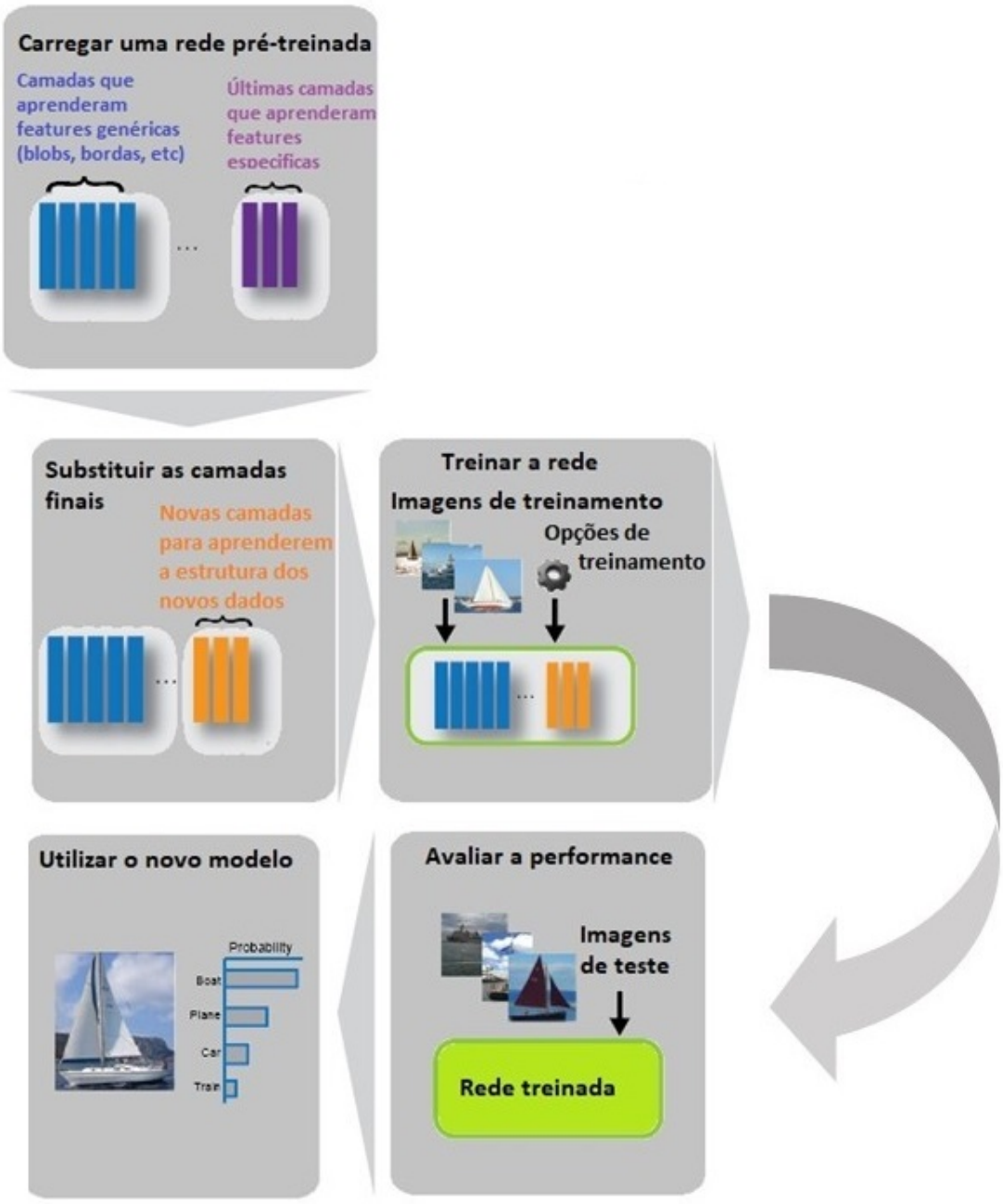
das convolucionais do modelo original, visto que estas são as responsáveis por extrair as feições que podem ser reaproveitadas para outras tarefas. Já as camadas totalmente conectadas, servem para realizar a classificação final e portanto não são úteis nesse cenário visto que as classes final são distintas.

Por exemplo, o modelo original da GoogLeNet foi treinada utilizando o conjunto ImageNet, que contém milhões de imagens de treinamento distribuídas em 1000 classes que incluem diversos animais, comidas e objetos comuns no dia-a-dia. Ao ser treinada utilizando tanta informação, a rede aprende a extrair uma infinidade de diferentes *features* para poder realizar a classificação entre as 1000 classes. Um típico esquema de transferência de aprendizado é apresentada na Figura 16.

Conforme apresentado na Figura 16, para realizar a transferência de aprendizado, carrega-se o modelo base a ser utilizado, substitui-se as camadas totalmente conectadas para que essas aprendam a estrutura do novo conjunto de dados, e a partir disso, treina-se e avalia-se a rede até que o resultado desejado seja atingido (SPRINGER, 2010).

Neste capítulo, foram apresentados os conceitos fundamentais para a compreensão dos capítulos posteriores. Na Seção 2.1 é mostrado o que são ataques de apresentação, como são realizados e quais os tipos existentes, bem como suas características. Em um segundo momento, é feita uma revisão sobre conceitos de redes neurais convolucionais, que são a base utilizada para os experimentos realizados nesta dissertação. Além dos conceitos sobre CNNs, também foram apresentadas as arquiteturas que foram empregadas como base para os testes feitos. Por fim, a técnica conhecida como transferência de aprendizado é apresentada visto que esta metodologia foi empregada para o treinamentos dos modelos comparados neste trabalho.

Figura 16: Típico fluxo de um treinamento utilizando transferência de aprendizado.



Fonte: (MATLAB 2018, r2018a)

3 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados trabalhos relacionados à área de detecção de ataques de apresentação, de forma a familiarizar o leitor com outros métodos presentes na literatura e com o estado-da-arte nesta área. Para realizar a busca pela literatura relacionada, foram feitas pesquisas nos repositórios *IEEExplore*, *ArXiv* e Elsevier, onde as principais *strings* de busca foram:

- *Face/Facial Spoofing Detection;*
- *Presentation Attack detection;*
- *Face Liveness Detection;*
- *Face Anti-spoofing*

Além dos resultados encontrados pela busca nos repositórios, também foram revisados artigos recomendados pelo colega de laboratório Lucas R. Schardosin, visto que este também trabalhou com detecção de ataques de apresentação durante seu doutorado. Como critério de inclusão dos artigos, levou-se em consideração o fator de impacto da revista ou conferência onde foi publicado, e os resultados (em termos de métricas conforme 4.3.2) apresentados pelo autor.

Os artigos pioneiros que descrevem estudos sobre contra medidas para ataques em sistemas biométricos datam de 2004 (LI et al., 2004). Durante a última década, um vasto número de estudo foi conduzido para explorar a vulnerabilidade de ataques direcionados a sistemas de reconhecimento facial, como em (ALBU, 2015; BENLAMOUDI et al., 2015). Em (DUC; MINH, 2009), o autor explorou brechas de segurança em métodos de controle de acesso baseados em reconhecimento facial em produtos das marcas Lenovo, Toshiba e Asus.

A partir dos tipos de ataques apresentados na Seção 2.1 e assumindo que existem diversas disparidades entre faces genuínas e artificiais, a análise de diversos fatores estáticos e dinâmicos tem sido empregados para a detecção deste tipo de ataque, dentre eles: textura, movimento, frequência, cor, formato e refletância. A ideia é que faces falsas apresentam distorções causadas pelo sistema de impressão (caso de ataques por fotos) ou pela tela do dispositivo (em ataques de vídeo). Além disso, também possuem menor qualidade, devido ao fato que as imagens falsas passaram por duas etapas de amostragem, uma durante a captura genuína da imagem do usuário e outra pelo dispositivo de reconhecimento facial, gerando uma diminuição na informação de alta frequência.

Devido a estas diferenças, a literatura apresenta métodos de detecção ataques de apresentação baseado nos seguintes fatores:

- **Baseado em texturas:** Assume diferenças entre texturas de faces genuínas e impostoras causada pelo processo de reamostragem do sinal (BHARADWAJ et al., 2013; SOLDERA et al., 2017).

- **Detecção de vivacidade:** Também conhecido como detecção de movimento, baseia-se no fato de que alguns ataques não apresentam sinais de vivacidade (do inglês *liveness*), ou seja, não apresenta piscadas ou movimentos faciais (ANJOS; CHAKKA; MARCEL, 2014).

- **Baseado em qualidade de imagem:** Similar ao método baseado em texturas, porém utilizam métricas de distorções de imagens, como variação de refletância, borramento e momento cromático (WEN; HAN; JAIN, 2015).

- **Baseados em outras pistas:** Estes métodos utilizam informações além das presentes em imagens 2D comum, como imagens em infravermelho, sensores de profundidade e voz (AKBULUT et al., 2017).

A Tabela 1 mostra os pontos positivos e as limitações de cada metodologia para detecção de ataques de apresentação.

Tabela 1: Comparação entre as vantagens e desvantagens entre metodologias.

Método	Pontos positivos	Limitações
Baseado em vivacidade	Boa generalização	Baixa robustez Elevado custo computacional
Baseado em texturas	Resposta rápida (<1s) Baixa complexidade computacional	Baixa generalização
Baseado em outras pistas	Alta robustez	Necessita equipamentos adicionais Resposta demorada (>3s)
Qualidade de imagens	Boa generalização Resposta rápida Baixa complexidade computacional	Classificadores diferentes para cada classe

Fonte: (WEN; HAN; JAIN, 2015)

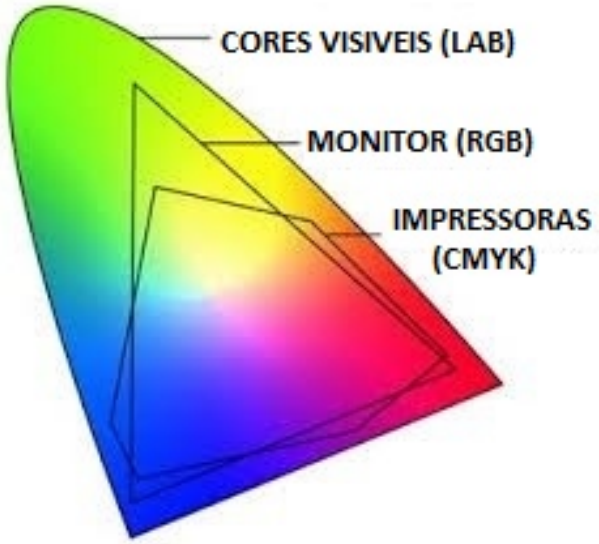
3.1 Detecção a partir de textura

Na literatura, as técnicas deste tipo são comumente conhecidos como métodos de análise de texturas visto que as propriedades apresentadas por um ataque podem ser consideradas como variações nas texturas capturadas pelo sensor.

O processo de recaptura da imagem, ou seja, utilizar a foto de uma foto, introduz disparidades na informação de cor que ajudam a distinguir entre uma face genuína e uma falsificada. Esta deformação de cor é oriunda da gama de cores apresentado pelas impressoras ou telas que produzem o face falsa para a tentativa de ataque (WEN; HAN; JAIN, 2015). Em geral, impressoras e telas possuem uma menor do que a paleta de cores visíveis, como mostrado na Figura 17, além disso, uma variação de cor também será introduzida dependendo do modelo da impressora ou tela sendo utilizada.

O espaço de cores "Lab" apresentado na Figura 17 representa um espaço definido pela comissão internacional de iluminação (CIE) em 1976 e expressa cores em três componentes: L para luminância, a e b para as componentes verde-vermelho e azul-amarelo, respectivamente. Este espaço foi desenhado de forma a obter uma percepção uniforme para um humano.

Figura 17: Demonstração da diferença de *gamut* entre as cores visíveis e as representáveis utilizando monitores RGB e impressoras CMYK.



Fonte: OKI Data Americas [1]

Já o espaço "CMYK" é um modelo de cores subtrativo usado em impressoras e seu nome tem origem nas quatro cores usadas para impressão: ciano, magenta, amarelo e preto (em inglês, cyan, magenta, yellow e key). Por ser um modelo subtrativo, seu funcionamento baseia-se no uso de tintas para diminuir a luz que é refletida pela superfície, tipicamente uma folha branca.

De acordo com (SOUZA et al., 2018), detecção a partir de texturas é o método mais empregado na literatura, contando com cerca de 69% dos trabalhos. Além das distorções apresentadas pelo processo de reamostragem, fotos impressas e apresentações de vídeos possuem padrões de textura que não ocorrem em faces naturais. Dentre as diversas abordagens utilizadas para realizar detecção de ataques de apresentação a partir de texturas, as que mais se destacam são: *Wavelets* de Gabor, informações de borda, histograma de gradientes orientados e padrões locais binários (do inglês *Local Binary Patterns*, ou LBP).

Dentre os métodos citados, o LBP é o mais utilizado em trabalhos recentes e é o que apresenta melhores resultados, como em (MAATTA; HADID; PIETIKAINEN, 2011), onde os autores apresentam taxas de classificação superiores a 95% no conjunto NUAA. O *Local Binary Patterns* é um tipo de descritor visual muito utilizado em tarefas de classificação baseados em visão computacional, foi desenvolvido por (AHONEN; HADID; PIETIKAINEN, 2006) e desde então tem sido bastante empregado em classificação baseado em texturas. O vetor de características extraídos pelo LBP é criado da seguinte maneira:

- Primeiramente, divide-se a região a ser examinada em células, por exemplo, 16×16 *pixels*.

- Para cada *pixel* na célula, compara-se com seus 8 vizinhos, iniciando com o ponto acima e a esquerda (*left-top*) e prosseguindo no sentido horário.

[1] Disponível em : <http://printsupervision.net/printer-color-matching-made-easy>. Acesso em: 3/5/2019

- Se o *pixel* central for maior que o valor do vizinho, escreva "0". Caso contrário, escreva "1". No final da comparação com os 8 vizinhos, irá formar um número binário com 8 bits.

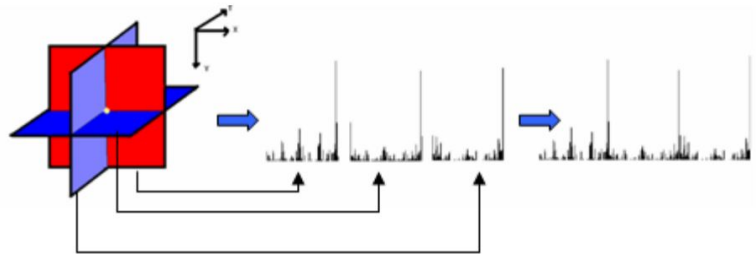
- Calcula-se o histograma dos valores encontrados dentro da célula, convertido para decimal. Este histograma irá resultar em um vetor de características de 256 dimensões.

- Por fim, normaliza-se o vetor e concatena-se os histogramas de todas as células, de forma a gerar um vetor de característica para toda a janela.

A partir do vetor de características extraído, é possível utilizar métodos de aprendizado de máquina como redes neurais ou Máquina de Vetor de Suporte (do inglês *Support Vector Machine*, ou SVM) para realizar a classificação.

Diferentes variações do método LBP são empregados em trabalhos de PAD, como (KOMULAINEN et al., 2013; BOULKENAFET; KOMULAINEN; HADID, 2015; SOUZA et al., 2017). Dos métodos que utilizam LBP, o LBP-TOP (LBP *from three orthogonal planes*), proposto por (FREITAS PEREIRA et al., 2013) se destaca por também levar em consideração as informações temporais presentes na sequência de vídeo, fazendo com o que este método atinja uma alta taxa de classificação correta no conjunto CASIA. Para isso, considera-se que cada série de quadros consiste de três planos ortogonais com interseção no centro do *pixel* na direção XY (LBP) normal, XT e YT, aonde T é o eixo do tempo. Considerando esta interseção entre os 3 planos, são gerados 3 histogramas que são concatenados conforme mostra a Figura 18.

Figura 18: Exemplo da interseção dos planos do LBP-TOP, bem como a concatenação dos histogramas gerados.



Fonte: (FREITAS PEREIRA et al., 2013)

Previamente, métodos que utilizavam o LBP original com SVM, obtinham taxas de erro médio (HTER) em torno de 15%. Ao utilizar estas informações temporais com o método LBP-TOP, o autor diminui esta taxa pela metade, atingindo 7.6% no conjunto de dados Replay-Attack (COSTA-PAZO et al., 2016).

3.2 Detecção baseada em movimento

Esta metodologia se baseia em detectar uma característica muito relevante para distinguir entre uma face viva e uma falsa: os movimentos realizados subconscientemente pelos órgãos e músculos presentes em uma face viva, como piscada de olhos (PAN et al., 2007), movimentos labiais (KOLLREIDER et al., 2007) e rotação da cabeça (BAO et al.,

2009). Para a detecção de movimentos de cabeça, estes métodos assumem que os movimentos apresentados por objetos planares, como fotos impressas e dispositivos móveis utilizados para ataques de vídeo, possuem movimentos totalmente diferente de objetos tridimensionais, como uma cabeça verdadeira. Dado que o movimento é uma característica relativa entre quadros do vídeo, estes métodos tendem a apresentar melhores resultados comparados à utilização de texturas.

No entanto, a utilização desta abordagem possui pontos negativos, como variação de comportamento entre diferentes usuários e o fato da frequência de movimento facial humana variar entre 0.2 e 0.5 Hz (BHARADWAJ et al., 2013). Esta frequência implica ser necessário capturar diversos segundos de vídeos para que seja possível observar estes movimentos, dificultando a utilização desta técnicas em casos onde seja necessário realizar a classificação em tempo real.

Em geral, a arquitetura destes algoritmos é composta por uma etapa de conversão para escalas de cinza. Esta imagem é então utilizada como entrada para um algoritmo de localização de faces e outro de estimação de fluxo óptico (FO). A saída destes dois métodos é então passada para um classificador que gera o resultado final, indicando se a sequência apresentada é genuína ou não. Nota-se que a saída da estimação de FO é um campo de velocidades κ que indica tanto a componente horizontal ($\vec{U_{ij}}$) quando a componente vertical ($\vec{V_{ij}}$) do movimento para cada *pixel* $(x, y) = (i, j)$.

O autor (ANJOS; MARCEL, 2011) propõe um algoritmo onde a direção do movimento é ignorada e utiliza somente a intensidade. A intensidade do movimento na face detectada e nas regiões de fundo são calculadas utilizando apenas médias das diferenças entre os quadros em escala de cinza, e uma normalização baseada em área que faz com que quadros com diferentes razões entre o tamanho da face e o fundo ainda possam ser comparados. Após esta etapa, são extraídas 5 características da imagem: O valor mínimo, máximo, médio e desvio padrão do sinal de intensidade de movimento. Além, também é utilizado a razão entre o somatório de todas as componentes não DC e DC, tomando como base a transformada de Fourier de N pontos.

Essas características extraídas, aliadas a um classificador treinado, permitem a avaliação da sincronização do movimento na cena. Se não houve movimento, indica um ataque com suporte físico, como um tripé segurando a apresentação. Se houver movimento demais, é muito provável que os dados de entrada sejam provenientes de um ataque de apresentação onde o impostor segura o dispositivo de ataque com as mãos. Acessos genuínos irão exibir movimentos descorrelacionados, uma vez que um usuário normal se move de maneira independente do fundo.

Como melhoramento do método descrito acima, (ANJOS; CHAKKA; MARCEL, 2014) propôs uma abordagem semelhante, denominada *Optical Flow Correlation*, ou OFC, porém, ao invés de utilizar as médias das intensidades entre quadros, é utilizado técnicas de estimação utilizando FO. Ao utilizar o fluxo óptico, espera-se obter uma estimação mais precisa dos parâmetros de movimento. O algoritmo OFC quantiza, gera histogramas, normaliza e compara diretamente os vetores de direção de movimento para gerar um *score* de correlação para cada quadro analisado. O algoritmo é dividido em duas etapas, descritas a seguir.

A primeira etapa toma como entrada a estimativa das velocidades horizontal e vertical do FO, além da demarcação da área que representa as faces (*Bounding boxes*). A partir destas entradas, ele realiza os seguintes passos com o objetivo de extrair características da cena:

- Primeiramente, a direção θ do movimento para cada *pixel* é calculada utilizando

as orientações verticais e horizontais do FO. As componentes de magnitude são descartadas, preservando apenas o movimento na direção θ_{ij} para cada ponto, onde $\theta_{ij} = atan2(V_{ij}, U_{ij})$, onde $atan2$ é definido abaixo pela Equação (2), e V_{ij} e U_{ij} representam as componentes verticais e horizontais, respectivamente:

$$atan2(y,x) = \begin{cases} tan^{-1}\frac{y}{x} & x > 0 \\ tan^{-1}\frac{y}{x} + \pi & y \geq 0, x < 0 \\ tan^{-1}\frac{y}{x} - \pi & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ \frac{-\pi}{2} & y < 0, x = 0 \\ indefinido & y = 0, x = 0 \end{cases} \tag{2}$$

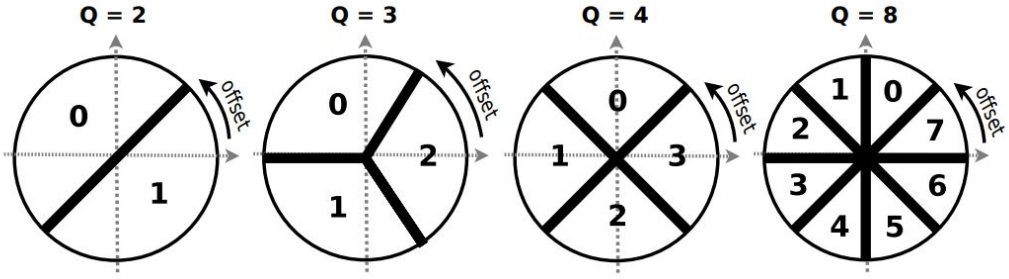
- A próxima etapa consiste em calcular histogramas normalizados para as regiões da face e de fundo, baseados nos ângulos encontrados na etapa anterior. O número de bins ($\mathcal{Q}$) é um hiperparâmetro do algoritmo, assim como o ângulo que deve iniciar a primeira pilha. Mais detalhes são apresentados na Figura 19.

- A distância χ^2 entre os histogramas da face e de fundo são calculados. Considerando F_i uma pilha i de um histograma da região da face e B_i a pilha correspondente no histograma da região de fundo, então a métrica χ^2 é definida seguindo a equação 3:

$$\chi^2(F,B) = \sum_i \frac{(F_i - B_i)^2}{F_i + B_i} \tag{3}$$

- Por último, é calculado a média dos valores encontrados de χ^2 para um conjunto de N quadros.

Figura 19: Exemplo da divisão dos bins e do *offset* para o ângulo inicial.



Fonte: (ANJOS; CHAKKA; MARCEL, 2014)

A segunda etapa do algoritmo consiste em utilizar estas características extraídas em um classificador binário que irá identificar a sequência como genuína ou impostora. Baseado na Equação (3), espera-se que quadros identificados como ataques possuam χ^2 próximo de 0, devido ao movimento correlacionado entre o fundo e a face. De forma contrária, ataques reais devem obter valores significativamente maiores que 0.

Utilizando esta metodologia, os autores apresentam resultados cujo erro médio varia entre 16.03% e 1.52% no conjunto de dados Replay-Attack, dependendo do conjunto de parâmetros selecionados

3.3 Detecção baseada em qualidade de imagem

Detecção baseada em qualidade de imagem, também chamada de detecção de vivacidade (do inglês *Liveness Detection*) é outra técnica utilizada para detectar ataques de apresentação em sistemas biométricos. Métodos de detecção de vivacidade são divididos em duas categorias primárias: I) *Baseados em Hardware*, quando algum dispositivo extra é adicionado além do sensor, de forma a estimar propriedades específicas, como pressão sanguínea, refletância dos olhos, etc. II) *Baseados em Software*, quando a distinção entre apresentações e genuínas e falsas são realizados após a captura da informação pelo sensor, utilizando algum algoritmo.

Estas abordagens se baseiam em que tentativas de acesso fraudulentas apresentam diversas características distintas de um acesso real. Alguns exemplos de métricas de qualidade utilizadas neste algoritmos são: grau de nitidez, nível de luminância, borramento, ruído, gradiente, covariância, entre outros.

O trabalho de (GALBALLY; MARCEL; FIERREZ, 2014b) propôs um método geral para avaliação de qualidade de imagens de íris, impressão digital e faces utilizando 25 métricas, porém, nenhuma destas métricas utiliza informações específicas da face. Já o trabalho de (WEN; HAN; JAIN, 2015) visa remediar esta falha, neste artigo, quatro feições são extraídas diretamente da face, sendo elas características de: reflexão especular, borramento, momento cromático e diversidade de cores. Estas características em conjunto com um classificador do tipo SVM, geram um termo de erro médio de 6.7% no *dataset* CASIA e 7.41% no REPLAY-ATTACK (COSTA-PAZO et al., 2016).

3.4 Detecção utilizando outras pistas

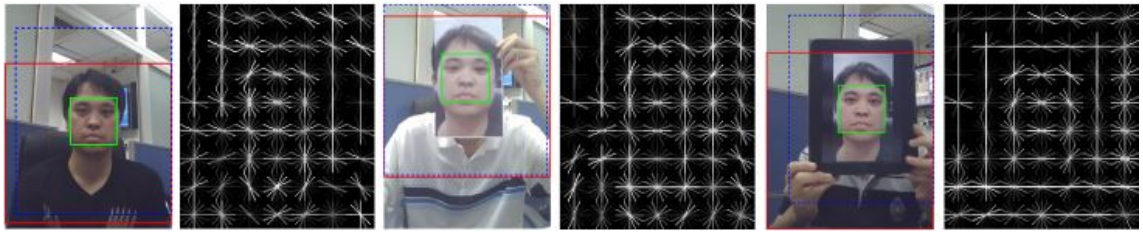
Devido a grande maioria dos trabalhos na área de detecção de ataques de apresentação apresentarem métodos baseados em textura, movimento ou vivacidade, todos os métodos restantes são classificados como "outras pistas". Dentre estes métodos, estão inclusos o uso de sensores extras além de câmeras tradicionais, como microfones, sensores de profundidade, câmeras infravermelhas, entre outros. Além disso, nesta categoria também se encontram os métodos chamados de interativos, onde é solicitado que o usuário realize uma série de atividade de forma a provar que é um humano, como realizar piscadas, sorrir, entre outros. Recentemente, também houve um surgimento do uso de CNNs nesta área de PAD devido a popularidade destes métodos em outras tarefas relacionadas a processamento de imagens e visão computacional.

Os autores (WANG et al., 2013), desenvolveram um método para a detecção de vivacidade usando uma reconstrução 3D da face utilizando uma única câmera. Neste trabalho, os autores constroem uma estrutura esparsa tridimensional da face do usuário baseado na correspondência de 68 *landmarks*. Após a construção do modelo, um SVM é utilizado para realizar a classificação.

Baseado na facilidade que um humano tem para detectar um ataque de *spoofing* baseado em algo suspeito na cena, como a presença de um *smartphone* ou foto impressa, (KOMULAINEN; HADID; PIETIKAINEN, 2013) propõe um método que busca informações no contexto geral da imagem, e não só na face. Para tal, é sugerido a utilização de uma classificação que utiliza o seguinte esquema: Se a parte superior do corpo não for localizada no *frame*, automaticamente classifica como impostor, caso contrário, é utilizado um conjunto de descritores do tipo Histograma de Gradientes (HoG) em uma área ao redor da face para tentar localizar o meio utilizado para realizar o ataque, visto que as

descontinuidades geradas pelo ataque de apresentação são notáveis neste descritor, conforme apresentado na Figura 20.

Figura 20: Exemplos das distorções causadas pelos meios de ataque nos descritores HoG.



Fonte: (KOMULAINEN; HADID; PIETIKAINEN, 2013)

De maneira similar, (SUN; HUANG; LIU, 2016) também propôs um método que consiste em buscar pistas do ambiente para auxiliar na classificação, porém, foi feito o uso de câmeras multi-espectrais que também capturam imagens no espectro do infra-vermelho próximo (do inglês *Near Infrared*, ou NIR). Para tal, o autor propõe dividir a imagem em 2 regiões: face e fundo, e a partir disso, buscar informações no contexto da imagem e correlação entre a iluminação e textura presentes nas duas regiões. Neste caso, imagens genuínas apresentam intensa reflexão especular nas extremidades, como nariz, testa e bochechas. Já folhas de papel A4 apresentam baixa refletância e telas de tablet possuem reflexões semelhantes em toda a sua superfície. Este método apresenta bons resultados, chegando a um termo de erro médio de 0.5% em casos de ataques de apresentação por vídeo, porém, necessita de equipamentos adicionais para realizar a classificação.

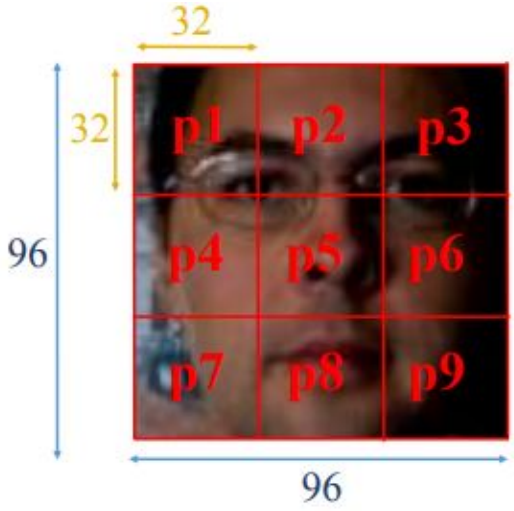
Recentemente, o uso de técnicas de aprendizado profundo também começou a ser explorado para a detecção de ataques de apresentação. Em (AKBULUT et al., 2017), o autor propõe dois métodos que fazem o uso de redes neurais convolucionais para realizar a classificação nos conjuntos Casia e NUAA. O primeiro método proposto é chamado de LRF-ELM e CNN. O modelo LRF-ELM é composto apenas por uma camada convolucional, uma camada de pooling e uma camada totalmente conectada. Já o método denominado CNN é composto cinco camadas convolucionais, três camadas totalmente conectadas. Após cada camada convolucional é utilizado uma camada de pooling e após cada totalmente conectada é utilizado dropout com probabilidade 0.5.

Além do modelo descrito, (SOUZA; PAPA; MARANA, 2018) também propõe uma nova arquitetura de rede neural convolucional chamada de lsCNN (do inglês *Locally Specialized CNN*) em conjunto com um novo método de treinamento. Este nova metodologia de treinamento tem como objetivo aprender feições faciais profundas locais baseados em duas etapas: (i) fase de treinamento para cada região pré-definida e fixa é utilizada para aprender tais feições; e (ii) uma etapa de ajuste fino para o modelo todo a partir dos pesos aprendidos independentemente em cada região, de forma a melhorar a generalização do modelo. A Figura 21 apresenta a divisão da face nas regiões pré-definidas.

A arquitetura da lsCNN é composta por 4 camadas de convolução seguidas de *pooling*, onde cada uma destas é seguida de uma etapa de normalização e aplicação da função de ativação ReLU (do inglês *Rectified Linear Unit*). Em seguida, o autor utiliza uma camada totalmente conectada para realizar a classificação entre as classes. A arquitetura mais detalhada é apresentada na Tabela 2. Esta abordagem atinge uma CCR de 95.56% no conjunto CASIA.

Já no contexto da utilização de transferência de aprendizado, em (SOUZA et al., 2018) é proposto uma metodologia onde a rede VGG-Face é utilizada para extrair as features das

Figura 21: Imagem de entrada para a lsCNN dividida em 9 regiões selecionadas.



Fonte: (SOUZA; PAPA; MARANA, 2018)

Tabela 2: Arquitetura da lsCNN

Camada	Tamanho do filtro	Passo	Mapa de entrada	Mapa de saída
Conv1	3×3	2	3×(96×96)	27×(94×94)
Pool1	2×2	2	27×(94×94)	27×(47×47)
Conv2	3×3	1	27×(47×47)	36×(45×45)
Pool2	2×2	2	36×(45×45)	36×(23×23)
Conv3	3×3	1	36×(23×23)	45×(21×21)
Pool3	2×2	2	45×(21×21)	45×(11×11)
Conv4	3×3	1	45×(11×11)	54×(9×9)
Pool4	2×2	2	54×(9×9)	54×(5×5)
FC1	-	-	54×(5×5)	1×(450)
Drop1	-	-	-	-
Softmax	-	-	1×(450)	1×(2)

Fonte: (SOUZA; PAPA; MARANA, 2018)

faces visto que essa é uma arquitetura que já foi treinada originalmente para a tarefa de reconhecimento facial. Utilizando as feições extraídas pela VGG-Face, é realizado o treinado de uma SVM para realizar a classificação dos quadros do conjunto Replay-Attack. Ao utilizar esta abordagem, o autor apresentou um resultado com taxa de classificação correta (CCR) de 93.95% neste conjunto, se aproximando do estado-da-arte, porém, com um tempo de treinamento reduzido.

A Tabela 3 apresenta uma comparação entre alguns métodos selecionados que foram apresentados anteriormente.

A partir da Tabela 3 nota-se um domínio de métodos que utilizam apenas a face como entrada, apesar de existirem abordagens como a HoG-Context (KOMULAINEN; HADID; PIETIKAINEN, 2013) que usam informações de ambiente a atinge bons resultados a partir disso. Já no contexto de uso de CNNs, nota-se que esta abordagem vem ganhando atenção nos últimos anos, porém, os autores focam apenas em desenvolver novas arquiteturas, sejam elas compactas ou profundas, ou em agilizar o processo de treinamento utilizando transferência de aprendizado. Visto que não foi encontrado nenhum trabalho

Tabela 3: Comparação entre métodos selecionados.

Método	Tipo	Entrada	Inovação
LBP-TOP	Textura	Face	Uso de informação temporal
HoG-Context	Outras pistas	Quadro	Uso de pistas dos arredores
LRF-ELM	CNN	Face	Modelo compacto e rápido
CNN	CNN	Face	Nova arquitetura
lsCNN	CNN	Face	Nova arquitetura
Transfer Learning	CNN	Face	Transf. de aprendizado com VGG-Faces

Fonte: Autor

relacionado onde o uso de informações dos arredores tenha sido explorada, decidiu-se explorar o efeito desta nova maneira de apresentar os dados no contexto de redes neurais convolucionais. Como para um humano a tarefa de detecção de *spoofing* torna-se simples ao observar o quadro inteiro, é possível que a rede também explore características como a presença de dedos nos arredores da foto, bordas provenientes dos dispositivos móveis utilizados.

O objetivo deste capítulo é a familiarização do leitor com trabalhos da área de detecção de ataques de apresentação facial. Para tal, primeiramente foi apresentado os quatro tipos de metodologias empregadas para a solução deste problema, sendo elas: baseada em texturas, em vivacidade ou movimento, em qualidade de imagem e baseado em outras pistas, onde as redes neurais convolucionais se encaixam. Durante cada seção, foram apresentados métodos selecionados ao longo da busca por literatura, de forma à apresentar trabalhos relacionados que são considerados estado-da-arte devido sua elevada taxa de classificação correta nos *datasets* da literatura. Por fim, é feita uma comparação entre alguns métodos, onde nota-se uma deficiência no uso de pistas do ambiente durante a classificação, e a partir disso, é gerada a motivação para este trabalho.

4 METODOLOGIA EXPERIMENTAL

Neste capítulo, primeiramente serão apresentados os materiais utilizados durantes os experimentos realizados, tais como o *Hardware* presente nos computadores utilizados e as bases de dados de ataques de apresentação. Em sequência, serão apresentados em detalhes a metodologia empregada para o treinamento das redes neurais convolucionais, bem como as métricas de avaliação utilizadas para medir o desempenho de cada modelo.

4.1 *Hardware* e ambiente de programação

Para a realização deste trabalho foram utilizados dois computadores disponíveis no Laboratório de Sinais e Instrumentação (LASIN) desta universidade. O primeiro computador consiste em um processador Intel Core i3-2100 com 32GB de memória RAM a 1666MHz e o segundo possui um processador Intel Core i5-2400 a 3.1GHz com a mesma configuração de memória RAM, ambos utilizam o sistema operacional Windows 7 Professional. Além disso, também foi utilizado um laptop contendo um processador Intel Core i7-4710MQ a 2.50GHz com 8GB de memória RAM a 1866MHz e uma placa gráfica nVidia GTX750M com 2GB de memória interna, esta máquina utiliza o sistema operacional Windows 10 Home. Apesar do laptop apresentar melhor desempenho no processamento de redes neurais devido a presença de uma GPU, ele só pode ser utilizado durante a etapa de teste visto que a quantidade de memória disponível é insuficiente para realizar o treinamento da rede. O uso de uma CPU para realizar estes experimentos acarreta em um tempo de processamento elevado, que apesar ser reduzido significantemente pelo uso de transferência de aprendizado, ainda resulta em cerca de 3h por época.

Em termos de software, todos os experimentos foram realizados utilizando MATLAB r2018a (MATLAB 2018, r2018a). Nos casos onde transferência de aprendizado foi utilizado, os modelos pré-treinados foram adquiridos utilizando a gerenciador de pacotes do próprio MATLAB, onde é possível realizar o *download* dos modelos originais treinados no conjunto ImageNet.

4.2 Datasets de ataques de apresentação

Nesta seção, serão apresentados os conjuntos de dados utilizados tanto para o treinamento das redes neurais convolucionais quanto para a avaliação do seu desempenho. A escolha destes conjuntos se a disponibilidade pública dos dados, além da vasta utilização deles na literatura, fazendo com que este trabalho possa ser facilmente comparado com o estado-da-arte. Além disso, também será apresentado o conjunto de dados contendo apenas faces, criado de forma a ser possível realizar uma comparação entre o uso de quadros

inteiros e apenas faces.

4.2.1 CASIA dataset

O primeiro *dataset* utilizado foi criado pelo instituto de automação da academia chinesa de ciências (CASIA) e foi disponibilizado pela primeira vez a partir da publicação de (ZHANG et al., 2012). Este conjunto de dados é composto de vídeos de diversos sujeitos, sendo que cada sujeito possui 8 vídeos em baixa resolução (640x480 *pixels*) e 4 em alta resolução (1280x720 *pixels*). Destes 12 vídeos por sujeito, 4 são de tentativas de acesso genuínos e os outros 8 são compostos de ataques de apresentação por foto impressa, foto distorcida e apresentação de vídeo. Os ataques que utilizam fotos foram feitos imprimindo quadros dos vídeos genuínos em papel A4 e os ataques de vídeo foram realizados utilizando estas mesmas imagens em um iPad.

A distribuição dos dados deste conjunto de dados é apresentada na Tabela 4, onde a denominação "G" e "I" representam genuíno e impostor, respectivamente.

Tabela 4: Distribuição dos dados do *dataset* CASIA.

	Sujeitos	Videos G	Videos I	*frames* G	*frames* I	Total
Treinamento	20	80	160	10914	34171	45085
Teste	30	120	240	15910	49897	65807
Total	50	200	400	26824	84068	110892

Fonte: (ZHANG et al., 2012)

Exemplos de quadros encontrados neste *dataset* podem ser vistos na Figura 22.

4.2.2 NUAA database

O conjunto intitulado *NUAA Photograph Impostor Database* (TAN et al., 2010) consiste em um *dataset* onde imagens foram coletadas utilizando diversas webcams baratas com o intuito de simular um ambiente de autenticação facial utilizando um laptop. O conjunto foi construído em 3 sessões com um intervalo de 2 semanas entre elas, sendo o local e a iluminação de cada sessão diferentes. No total, 15 sujeitos participaram da captura dos dados, e foi coletado 500 quadros de cada sujeito. Durante a captura, foi pedido para que os sujeitos olhassem frontalmente para a câmera, com uma expressão neutra e sem movimentos nem piscadas. Este requerimento foi feito para que as faces genuínas se aproximassem o máximo possível de fotos impressas.

Para os ataques de apresentação, foi tirado uma foto em alta definição utilizando uma câmera profissional Canon de forma que a face cobrisse pelo menos $2/3$ da área da fotografia. Após este processo, as fotos foram impressas em uma impressora HP em 2 tipos de papel, sendo eles: papel fotográfico comum de tamanho 6.8x10.2cm e 8.9x12.7cm, e em papel A4 70g. A distribuição dos dados encontrados neste conjunto de imagens é apresentado na Tabela 5

Exemplos de imagens capturadas neste conjunto são apresentadas na Figura 23.

4.2.3 Dataset de faces

Para poder testar a hipótese de que a utilização do quadro inteiro é de fato vantajoso, foi necessário treinar também uma rede neural com a mesma arquitetura, porém, utilizando apenas as informações faciais. Para tal, as faces presentes em todos os quadros tanto do *dataset* NUAA quanto do CASIA foram extraídas utilizando o algoritmo de lo-

Figura 22: Exemplos de quadros do *dataset* CASIA para o sujeito número 7. Nota-se que algumas imagens podem apresentar distorções devido ao processo de redimensionamento para a resolução de entrada de 224×224 das CNNs.

(a) Quadro genuíno;

(b) Quadro genuíno de alta resolução;

(c) Ataque de foto impressa;

(d) Ataque de foto impressa deformada.

(e) Foto impressa com recorte nos olhos;

(f) Apresentação de vídeo.

Fonte: (ZHANG et al., 2012)

calização facial proposto por (VIOLA; JONES, 2001), que é o método mais utilizado na literatura para extração de faces.

Como, o algoritmo de detecção facial não possui 100% de taxa de acerto, não foi possível extrair todos os quadros de ambos os *dataset*s, porém, foram encontrados 48576 quadros para treinamento e 74930 de teste, o que representa cerca de 95% do total de imagens.

Exemplos das faces extraídas dos conjuntos originais é mostrada na Figura 24.

4.3 Experimentos realizados

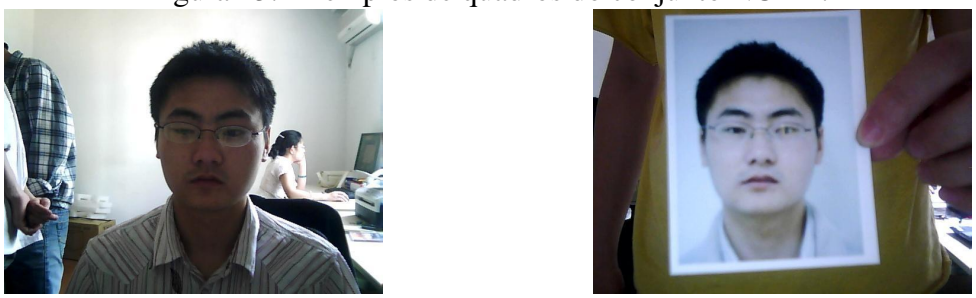
Nesta seção, serão apresentados a metodologia que foi usada pra o treinamento dos modelos sendo avaliados e as métricas utilizadas para avaliar o desempenho obtido. A Subseção 4.3.1 apresenta o protocolo de divisão dos dados para treinamento e testes, os intervalos de buscas dos hiperparâmetros testados, bem como uma regra para descartar de forma precoce modelos com baixo desempenho. Já a subseção 4.3.2 apresenta as princi-

Tabela 5: Distribuição dos dados do *dataset* NUAA.

	Sessão 1	Sessão 2	Sessão 3	Total
Treinamento				
Genuíno	889	854	0	1743
Impostor	855	893	0	17483
Total	1744	1747	0	3491
Teste				
Genuíno	0	0	3362	3362
Impostor	0	0	5761	5761
Total	0	0	9123	9123

Fonte: (TAN et al., 2010)

Figura 23: Exemplos de quadros do conjunto NUAA.



(a) Quadro genuíno; (b) Ataque de foto impressa;

Fonte: (TAN et al., 2010)

pais métricas utilizadas na literatura, que também foram empregadas nesta dissertação.

4.3.1 Treinamento das CNNs

Neste contexto de redes neurais, treinamento se refere ao processo de determinar os parâmetros ideais que resultarão no melhor desempenho do modelo (Google, 2019). Para realizar a transferência de aprendizado para as CNNs que foram selecionados, primeiramente é necessário particionar os *datasets* em um conjunto de treinamento e outro de testes. Em geral, durante o treinamento de redes neurais, é comum distribuir o conjunto de dados em 70% para treinamento e 30% para teste para que seja possível avaliar o desempenho do modelo com dados que não foram utilizados durante o treinamento. No entanto, tanto o conjunto CASIA quanto o NUAA possuem protocolos próprios de divisão, onde cada *dataset* especifica as porções dos dados que devem ser utilizados tanto para treinamento quanto para teste. Para manter este trabalho consistente com os outros métodos da literatura, os seguintes protocolos propostos por cada conjunto são utilizados:

- Para o conjunto **CASIA**, o protocolo define a utilização de imagem de 20 sujeitos selecionados para treinamento e os outros 30 para teste. Resultando em 45085 amostras (40%) de treinamento e 65807 (60%) para testes.

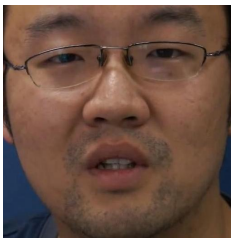
- No conjunto **NUAA**, é definido a utilização de 3491 (27%) amostras para o treinamento e 9123 (73%) para testes.

Para explorar a proposta de que a utilização do quadro completo é vantajoso devido a presença de informações do ambiente, foram treinados modelos utilizando apenas as faces

Figura 24: Exemplos de quadros do conjunto de dados criados extraindo apenas as faces.

(a) Face genuína NUAA;

(b) Face impostora NUAA;



(c) Face genuína CASIA;

(d) Face impostora CASIA.

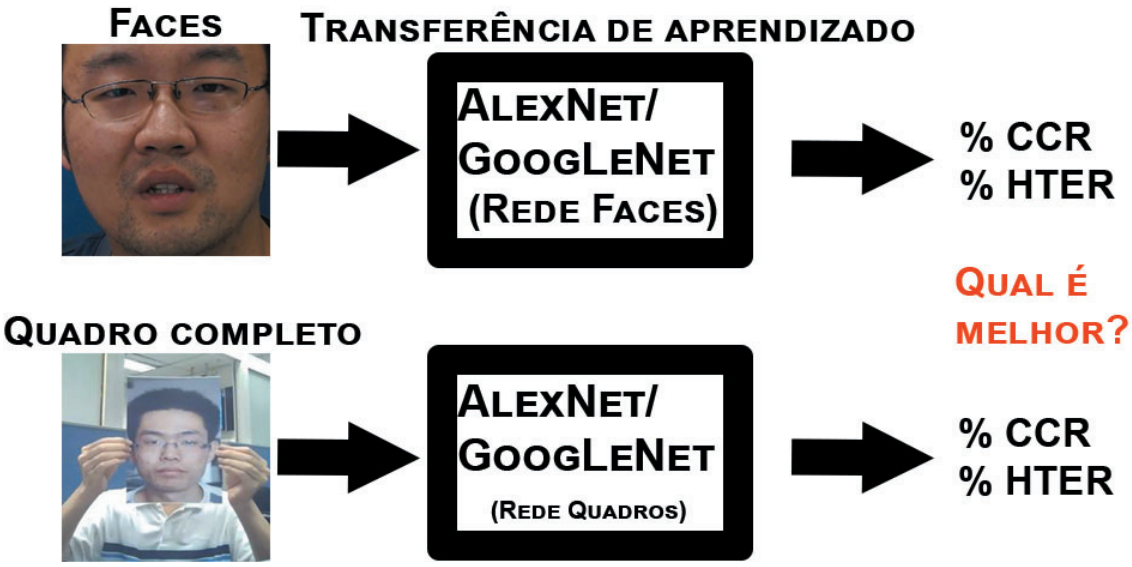
Fonte: Autor

do conjunto de faces introduzido na Subseção 4.2.3, que serão chamados de *Rede Faces*, e também foram desenvolvidos modelos utilizando os quadros completos dos conjuntos CASIA e NUAA, que serão referenciados como *Rede Quadros*. A Figura 25 apresenta uma ilustração simplificada deste processo.

Para realizar a transferência de aprendizado para as arquiteturas propostas, primeiramente o modelo desejado é obtido a partir do gerenciador de pacotes do MATLAB, onde é baixado um arquivo contendo a arquitetura, os pesos e as conexões da rede conforme obtidos pelos autores originais após realizarem treinamento no conjunto ImageNet. Após, todas as camadas totalmente conectadas são retiradas da rede, visto que estas camadas são responsáveis por realizar a classificação a partir das feições extraídas pelas camadas convolucionais, e portanto devem ser retreinadas. Em seguida, novas camadas totalmente conectadas (TC) são adicionadas à arquitetura, sendo essas inicializadas com pesos aleatórios seguindo uma distribuição normal. A quantidade de camadas adicionadas e o número de neurônios em cada uma foi um dos parâmetros testados durante os experimentos. A única exceção fica por conta do tamanho da última camada que deve, obrigatoriamente, ter 2 neurônios, que representam o número de classes, genuíno e impostor.

Durante a etapa de treinamento, foi realizada uma busca do tipo Grid Search para selecionar os melhores hiperparâmetros para cada arquitetura, os parâmetros ajustados desta maneira e seus intervalos de busca são mostrados na Tabela 6. Nota-se que neste contexto de treinamento de redes neurais, ”mini-lote” se refere ao número de amostras usadas em cada etapa de atualização dos parâmetros do modelo, e uma ”época” (do inglês *epoch*) se completa quando todos os dados do conjunto de treinamento são propagados pela rede. Por exemplo, em um conjunto de dados com 1000 amostras, pode-se utilizar um mini-lote de 100 amostras, o que implica que uma época será completada após a passagem dos 10 mini-lotes pela RNA. O uso de mini-lotes ao invés de realizar o treinamento com o conjunto completo de uma só vez é motivado principalmente por: primeiramente, atualiza-se os pesos das conexões da rede com mais frequência, além disso, nem sempre é possível carregar todo o conjunto de treinamento na memória de uma só vez, o que obriga o uso de pequenos lotes (Google, 2019).

Figura 25: Ilustração da comparação proposta.



Fonte: Autor

Tabela 6: Hiperparâmetros testados e seus intervalos de busca.

Parâmetros	Valor mínimo	Intervalo	Valor máximo	N°
Taxa de aprendizado	$1e^{-2}$	$1e^{-1}$	$1e^{-5}$	4
Tamanho do mini-lote	16	$2^n, n = 4, ..., 8$	256	5
Número de épocas	1	1	10	10
Número de camadas TC	1	1	3	3
Número de neurônios	16	$2^n, n = 4, ..., 11$	2048	8

Fonte: Autor

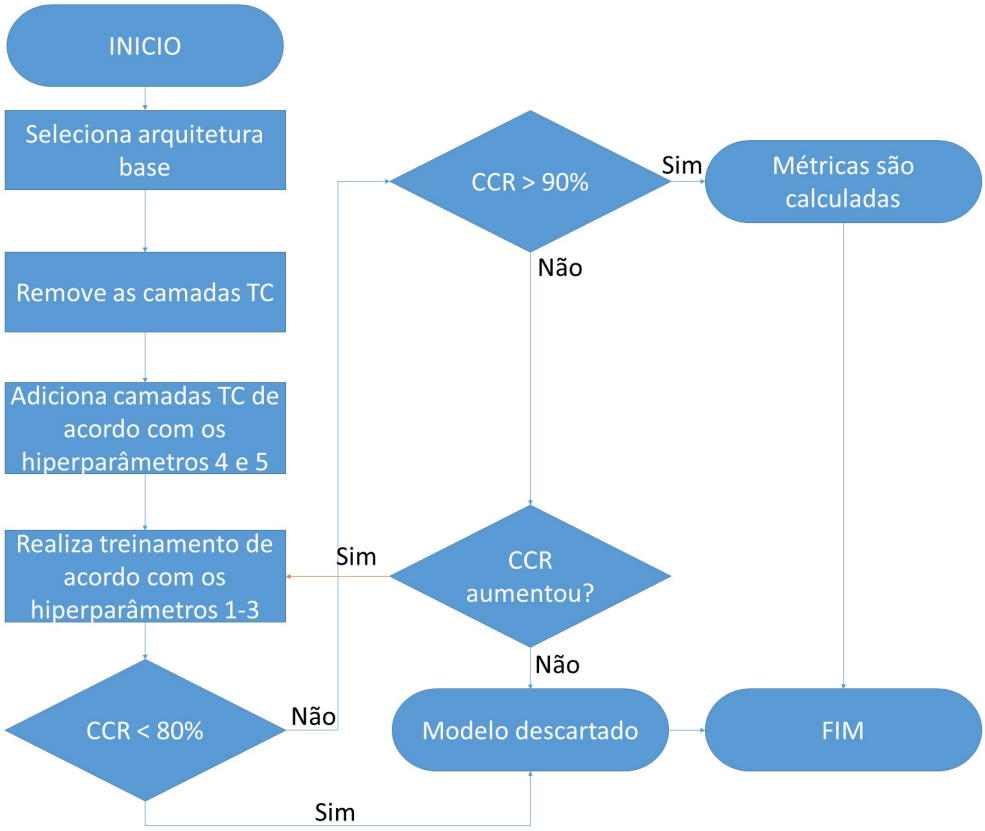
A partir do número de possibilidades para cada parâmetro presentes na Tabela 6, nota-se que existem 4800 combinações distintas para serem exploradas. Considerando que o número de épocas varia entre 1 e 10, e existem 480 combinações para serem avaliadas por esse determinado número de iterações, tem-se que é necessário realizar 21600 épocas de treinamento para cada arquitetura sendo testada.

Com base no número de combinações possíveis, notou-se que o tempo necessário para realizar todos os experimentos seria inviável com os recursos computacionais e com o tempo disponível. Para reduzir o tempo gasto com treinamento, adotou-se o seguinte protocolo de avaliação: Ao final de cada época de treinamento com um conjunto de parâmetros do *Grid Search*, é avaliada a taxa de classificações corretas (CCR, do inglês *Correct Classification Rate*), no conjunto de testes, e o modelo é descartado ou não de acordo com a seguinte regra:

- Se a CCR é menor que 80%, a arquitetura é descartada;
- Se a CCR é maior que 80% mas menor que 90%, o treinamento continua;
- Se a CCR for maior que 90%, as métricas de avaliação utilizadas na literatura são calculadas, e posteriormente o modelo é submetido a um ajuste fino.

Em conjunto com o protocolo acima, o modelo também pode ser descartado caso sua CCR não obtenha um aumento mínimo de 1% em comparação com a última época. Ao final deste procedimento, as melhores redes de cada arquitetura são selecionadas para comparação com outros métodos do estado-da-arte. Nota-se que esta etapa foi realizada duas vezes, uma para o treinamento da rede treinada apenas com imagens das faces e outra para a rede que recebe o quadro inteiro. Um fluxograma deste esquema de treinamento é mostrado na Figura 26.

Figura 26: Fluxograma do esquema de treinamento.



Fonte: Autor

4.3.2 Métricas de avaliação

Após a seleção dos modelos como mostrados anteriormente, é feita a etapa de testes, onde o conjunto de dados separados previamente para este objetivo é dado como entrada para o modelo treinado e seu desempenho é avaliado. A partir das classificações realizadas neste conjunto, são então calculadas as métricas as quais aparecem com frequência em trabalhos de detecção de ataques de apresentação (AKBULUT et al., 2017; MANJANI et al., 2017; LUAN et al., 2017), são elas:

- False Acceptance Rate (FAR) - A razão de quadros impostores que foram classificados como genuínos;
- False Rejection Rate (FRR) - A razão de quadros genuínos que foram classificados como impostores;

- Half-Total Error Rate (HTER) - A média entre o FAR e o FRR.

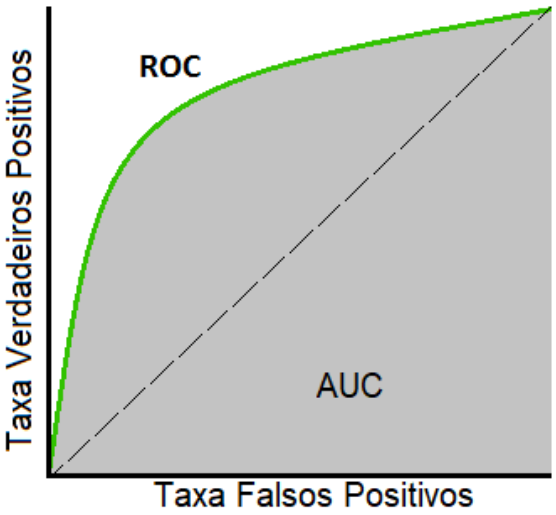
A utilização desta métrica ao invés de empregar apenas a taxa de classificação correta é interessante por alguns motivos. Primeiramente, no caso de classes não balanceadas, a CCR elevada pode indicar um *bias* para a classe que possui um número maior de ocorrências. Além disso, um modelo que possua um elevado número de falsos positivos ou falsos negativos não é interessante, por exemplo, um modelo que identifique todos os impostores não é interessante caso ele rejeite boa parte das classes genuínas neste processo.

Além destas métricas, também é comum a utilização da curva de característica de operação do receptor (curva ROC, do inglês *Receiver Operating Characteristic*). Esta curva é utilizada para selecionar uma regra de predição, ou seja, o valor do limiar $\hat{\pi}$ que irá separar as classes. A partir disso, realiza-se a classificação conforme Equação (4), onde Y representa a classificação final e y a saída do classificador, que será entre 0 e 1.

$$Y = \begin{cases} 1 & y \geq \hat{\pi} \\ 0 & y < \hat{\pi} \end{cases} \tag{4}$$

A curva ROC é encontrada plotando a taxa de verdadeiros positivos vs taxa de falsos positivos para diferentes valores de $\hat{\pi}$ e partir disso é possível escolher o melhor limiar de forma a obter um balanço entre os verdadeiros positivos e os falsos positivos. Em conjunto, também costuma-se calcular a área sob a curva ROC (AUC, do inglês *Area Under Curve*), visto que essa métrica representa o grau de separabilidade entre as classes, ou seja, mostra o quanto o modelo consegue distinguir entre as classes. Quando maior a área, melhor a separação entre as classes, sendo que uma área igual a 1 representa um classificador perfeito. Um exemplo de curva ROC é mostrado na Figura 27.

Figura 27: Exemplo de curva ROC. A linha pontilhada representa a linha de não-discriminação e a região em cinza a área sob a curva.



Fonte: Towards Data Science [1]

A partir da Figura 27, nota-se que caso o limiar selecionado seja muito baixo, o resultado será um aumento no número de amostras que serão classificadas como positivas, que

[1] Disponível em : <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5?gi=a7364d56d6bb>. Acesso em: 3/5/2019

acarretará tanto no aumento de verdadeiros positivos, quanto no número de amostras negativas sendo classificadas erroneamente, aumentando também a taxa de falsos positivos.

Na literatura de ataques de apresentação, também é comum a utilização da chamada curva DET (MARTIN et al., 1997) (do inglês *detection error tradeoff*), que é similar a curva ROC, porém, na curva DET é feito a plotagem da taxa de falsos positivos (FRR) pela taxa de falsos negativos (FAR). A partir desta curva DET, procura-se o ponto cujo limiar implica em $FAR = FRR$ e este ponto é utilizado para reportar os resultados finais do classificador.

Neste capítulo, foi apresentado o *Hardware* utilizado durante os experimentos, de forma a apontar o uso de apenas uma CPU no treinamento, que resulta em um tempo elevado para esta etapa do desenvolvimento do trabalho. Em sequência, são apresentados os conjuntos de dados utilizados para os experimentos utilizando quadros inteiros, bem como a metodologia empregada para a construção de um novo conjunto contendo apenas faces para que fosse possível a comparação entre o método tradicional e o proposto nesta dissertação. A escolha destes conjuntos se deve ao fato de serem disponíveis publicamente, sem a necessidade de licenças de uso, além do extensivo uso destas bases em outros trabalhos da área. Por fim, é introduzido o modo como os modelos foram treinados, os hiperparâmetros testados e seus intervalos de busca, bem como as métricas utilizadas para a avaliação de cada modelo. Baseado número de combinações possíveis de parâmetros de ajuste, notou-se a necessidade de uma regra para que modelos com baixo desempenho inicial fossem descartados rapidamente, de forma a reduzir o tempo total de treinamento.

5 RESULTADOS EXPERIMENTAIS

Neste capítulo, serão apresentados os resultados obtidos nos experimentos realizados, e os mesmos serão comparados à outras abordagens presentes na literatura que são considerados estado-da-arte. Além disso, será feita uma comparação entre os resultados obtidos utilizando o quadro inteiro e o usando apenas das faces. Para realizar uma comparação completa entre os modelos, tanto os que foram treinados somente com as faces quanto os que utilizam quadros completos foram analisadas em ambos os conjuntos de dados CASIA (ZHANG et al., 2012) e NUAA (TAN et al., 2010) e seus resultados em termos de métricas de avaliação foram analisados.

Primeiramente, foram realizados experimentos para avaliar o desempenho as arquiteturas das CNNs escolhidas para a transferência de aprendizado. Para tal, diversos modelos utilizando como base tanto a AlexNet quanto a GoogLeNet foram treinados fazendo uso da metodologia apresentada na Subseção 4.3.1, onde a rede treinada previamente no conjunto ImageNet é reutilizada para detecção de ataques de apresentação facial. Devido a utilização de uma CPU para o treinamento ao invés de uma GPU, o tempo necessário para realizar cada época se torna elevado, em torno de 200 minutos na AlexNet e 300 minutos na GoogLeNet. Após realizar diversos experimentos, os melhores resultados encontrados por cada arquitetura são mostrados na Tabela 7. Para facilitar o entendimento, foram apresentados apenas os resultados das CNNs que obtiveram o melhor desempenho nas métricas de avaliação conforme Seção 4.3.2.

Tabela 7: Melhores resultados encontrados por cada arquitetura utilizando diferentes combinações de *datasets*.

Arquitetura	*datasets* utilizados	CCR (%)
AlexNet	Casia + NUAA	88
GoogLeNet	NUAA	81.2
GoogLeNet	Casia	90
GoogLeNet	**Casia + NUAA**	**96.74**

Fonte: Autor

A baixa CCR da GoogLeNet utilizando apenas o conjunto NUAA pode ser explicado pelo tamanho reduzido desta base. A quantidade de amostras contido nele é insuficiente para que a rede ajuste seus parâmetros de forma adequada, o que causa um efeito de *overfitting*. Já o resultado abaixo de 90% da AlexNet é explicado de maneira semelhante, esta arquitetura possui cerca de 60 milhões de parâmetros a serem ajustados durante o treinamento, com isso, um *dataset* de pouco mais de 110 mil amostras é insuficiente mesmo utilizando a técnica de transferência de aprendizado. Este efeito também pode ser inferido pelo gráfico de treinamento mostrado na Figura 28. Na Figura 28, o gráfico

superior representa a CCR, onde a linha contínua e a linha pontilhada representam o conjunto de treinamento e teste, respectivamente. Já o gráfico inferior representa o valor da função custo para o conjunto de treinamento e teste seguindo a mesma lógica.

A partir da Figura 28, é possível notar que o valor da função custo para o conjunto de treinamento atinge um valor próximo de 0 após poucas iterações, o que não se repete no conjunto de testes, indicando que a rede está apenas ”decorando” o conjunto de treinamento. Este resultado pode ser confirmado a partir da parte superior da mesma figura, onde a CCR para o conjunto de treinamento é elevada enquanto nos testes fica em torno de 80%. Com bases nos resultados insatisfatórios, decidiu-se descartar a arquitetura AlexNet e prosseguir os experimentos apenas com a GoogLeNet.

Para explorar a proposta de que a utilização do quadro completo é vantajoso devido a presença de informações do ambiente, primeiramente foi realizado uma comparação entre o melhor resultado encontrado treinando um modelo com quadros inteiros, *Rede Quadros* e o melhor utilizando apenas as faces extraídas conforme apresentado na Subseção 4.2.3, *Rede Faces*. Ambos modelos foram baseados na arquitetura GoogLeNet devido ao melhor desempenho como apresentado na Tabela 7 e utilizaram como função custo a entropia cruzada, apresentada na equação (5).

$$\mathcal{L} = -(y.log(\hat{y}) + (1-y)log(1-\hat{y})) \tag{5}$$

Nesta função custo, y representa a classe correta para a observação o, $\hat{y}$ é a predição feita pelo modelo e log indica o logaritmo natural. Os conjuntos de hiperparâmetros de treinamento que obtiveram os melhores resultados tanto para a Rede Quadros tanto para a Rede Faces são apresentados na Tabela 8.

Tabela 8: Melhores hiperparâmetros para cada arquitetura testada.

Hiperparâmetros	Rede Faces	Rede Quadros
Taxa de aprendizado inicial	$1e^{-3}$	$1e^{-3}$
Tamanho de mini-lote	64	64
Número de épocas	10	2
Número de camadas TC	1	1
Número de neurônios	1024	1024

Fonte: Autor

Com base no treinamento realizado segundo descrito anteriormente e com os hiperparâmetros apresentados na Tabela 8, a Tabela 9 apresenta uma comparação entre a CCR encontrada realizando uma avaliação completa entre os modelos em ambos os conjuntos de dados. Ou seja, tanto a Rede Faces quanto a Rede Quadros foi avaliada no *dataset* de faces e quadros.

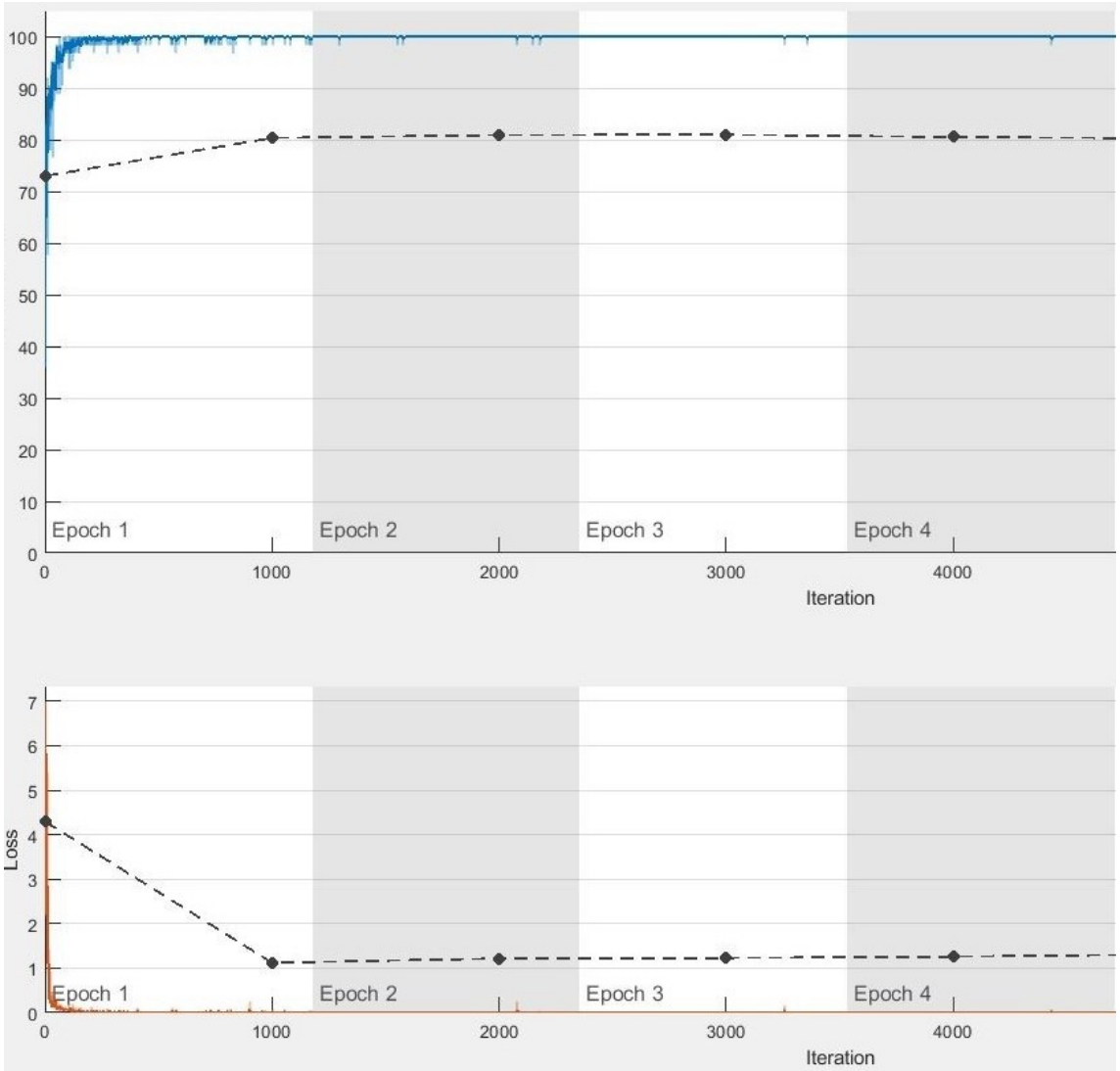
Tabela 9: Taxas de CCR em um teste de *datasets* completo.

Modelo	*Dataset* de faces	*Dataset* de quadros inteiros
Rede Face	**95 %**	23.9 %
Rede Quadros	94.47%	**96.74 %**

Fonte: Autor

A partir da Tabela 9, nota-se que de fato é vantajoso a utilização dos quadros completos. Durante um primeiro olhar, um ganho de pouco menos de 2% pode não parecer tão

Figura 28: Gráfico de treinamento para a arquitetura AlexNet gerada automaticamente pelo MATLAB durante o treinamento quando a função *trainNetwork* do pacote *Deep Learning Toolbox*. Nota-se que devido a ser um gráfico gerado automaticamente, suas legendas se encontram em inglês. Glossário: ”epoch” - época; ”Loss” - valor da função custo; ”iteration” - Iteração.



Fonte: Captura de tela da ferramenta de treinamento de CNNs do pacote *Deep Learning Toolbox* do MATLAB

significante, porém, como a taxa de 95% já é considerado alta precisão, portanto, um ganho de 2% é interessante. Além disso, a rede quadros apresentou desempenho semelhante a rede faces quando avaliada no *dataset* faces e também exige uma etapa a menos, visto que não é necessário extrair a face do quadro antes de usá-lo como entrada na CNN.

Para explorar com mais profundidade esta vantagem da Rede Quadros, os mapas de ativações das camadas convolucionais foram analisadas em mais detalhes. Como estas são as responsáveis por extrair as características que serão utilizadas para a classificação entre genuína e impostora, ao observar os mapas de ativações, é possível compreender melhor o funcionamento da rede, por exemplo, checar quais regiões da imagem que estão gerando altos valores de ativação.

Para ilustrar este processo, os mapas de ativações das camadas convolucionais foram sobrepostas na imagem de entrada original, de maneira a tornar possível a visualização das regiões que a rede neural está focando. Nestas figuras, a cor verde representa regiões com baixas ativações, em contrapartida, áreas pintadas de rosa indicam locais de importância para a CNN, resultando em altas ativações. As Figuras 29, 30 e 31 representam os mapas para os sujeitos 2, 7 e 13, respectivamente, do conjunto de testes do *dataset* CASIA.

As Figuras 29(a-b), 30(a-b) e 31(a-b) apresentam exemplos de mapas de ativação para uma tentativa de acesso genuíno. A partir destas figuras, nota-se que neste caso a CNN apresenta um foco maior em feições faciais, porém, também apresenta altas ativações na roupa do usuário, como na Figura 31(b). Também nota-se que mesmo quando 2 pessoas estão presentes na cena, a rede neural tende a focar na face mais ao centro e que ocupa a maior porção do quadro, conforme visto na Figura 31(a).

As Figuras 29(c-d), 30(c-d) e 31(c-f) ilustram casos onde é feito a tentativa de um ataque de apresentação utilizando foto impressa. Como apresentados nas Figuras 30(c) e 31(c), nota-se uma recorrência no uso de feições faciais, no entanto, como pode ser visto nas Figuras 29(c-d), 30(d) e 31(d-e), seções referentes ao corpo do impostor, como dedos, braços e mãos também geram altos valores de ativação.

Já as Figuras 29(e-f) e 30(e-f) mostram casos onde é realizado um ataque por vídeo. Nestas situações, percebe-se um comportamento semelhante ao caso com fotos impressas, ou seja, a rede utiliza feições faciais combinadas com informações dos arredores, neste caso, a tela do dispositivo móvel sendo utilizado para apresentar o vídeo.

A partir dos mapas de ativação apresentados, pode-se concluir que a CNN de fato faz o uso de informações do ambiente para aperfeiçoar sua classificação. Após obter resultados superiores utilizando a *rede quadros* e confirmar a utilização de pistas adicionais a partir dos mapas de ativação, decidiu-se descartar a *rede faces* e prosseguir a utilização somente dos quadros completos. Baseado nisso, foi realizado um teste de *dataset* cruzado, que são frequentemente utilizados na literatura. Neste tipo de teste, o modelo é treinado em um conjunto de dados e testado em outros para avaliar de forma mais completa a generalização dele para diferentes cenários que não foram vistos durante o treinamento. O resultado deste experimento pode ser visto na Tabela 10.

Figura 29: Mapas de ativação para o sujeito 2 do conjunto CASIA.

(a) Alta ativação em feições faciais;

(b) Alta ativação no rosto;

(c) Alta ativação nos dedos;

(d) Alta ativação nas mãos do impostor.

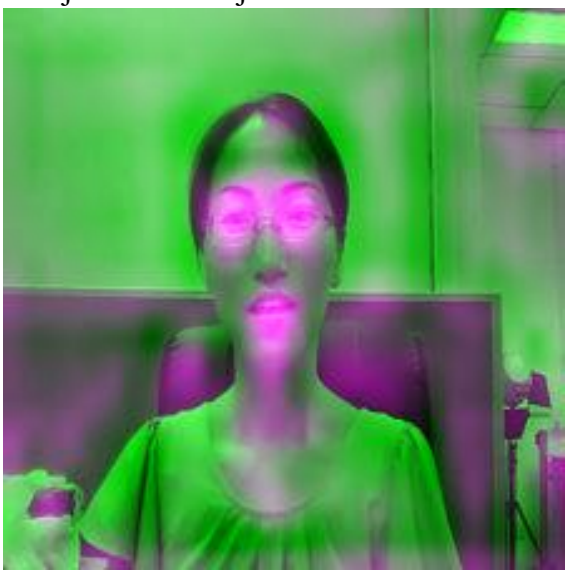
(e) Alta ativação nas bordas;

(f) Alta ativação ao redor do dispositivo.

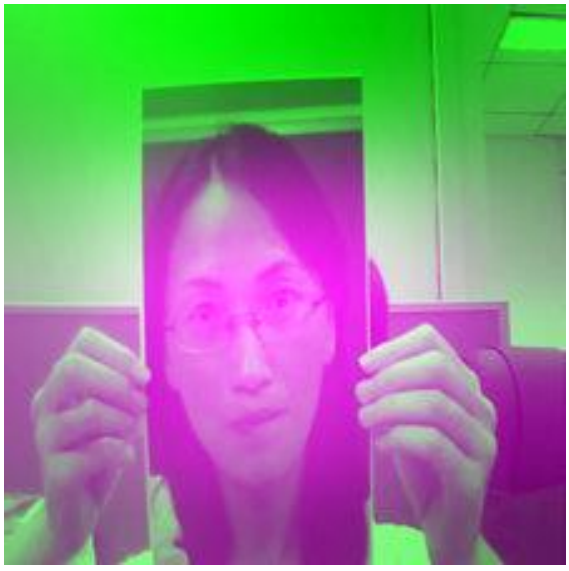
Fonte: Autor.

Figura 30: Mapas de ativação para o sujeito 7 do conjunto CASIA.

(a) Alta ativação em feições faciais;



(b) Alta ativação em regiões salientes;



(c) Alta ativação no rosto da foto;

(d) Alta ativação nas mãos do impostor.

(e) Alta ativação na face apresentada;

(f) Alta ativação ao redor do tablet.

Fonte: Autor.

Figura 31: Mapas de ativação para o sujeito 12 do conjunto CASIA.

(a) Alta ativação no rosto;

(b) Alta ativação na roupa;

(c) Alta ativação em feições faciais;

(d) Alta ativação no dedo do impostor.

(e) Alta ativação nos braços;

(f) Alta ativação nas mãos do impostor.

Fonte: Autor.

Tabela 10: Comparação de desempenho utilizando teste de *dataset* cruzado.

Testado em	CCR (%)	FAR (%)	FRR (%)	HTER (%)
Treinado NUAA				
NUAA	98.36	2.49	1.48	1.32
Casia	82.47	15.87	22.71	19.29
Treinado Casia				
NUAA	92.29	11.71	0.83	6.2748
Casia	98.72	1.63	0.13	0.88

Fonte: Autor

Como notado na Tabela 10, as redes possuem desempenho excelente quando testadas no conjunto de testes do mesmo *dataset* em que foram treinadas, porém, o desempenho reduz drasticamente no teste cruzado. Esta diferença pode ser explicadas pelas diferentes características de cada conjunto, como iluminação, ângulo e afastamento entre o usuário e a câmera. Para tentar reduzir estes efeitos e construir um modelo mais robusto a estas diferenças, foi feito o treinamento de um modelo combinando os dois conjuntos de treinamento, formando assim um *dataset* maior e mais completo. Os resultados encontrados e as curvas ROC para este modelo são apresentadas na Tabela 11 e Figura 32, respectivamente.

Tabela 11: Resultados obtidos após junção dos conjuntos de treinamento.

dataset testado	CCR (%)	FAR (%)	FRR (%)	HTER (%)
Treinado Casia + NUAA				
NUAA	97.61	3.5	0.47	1.99
Casia	98.81	1.56	0	0.78

Fonte: Autor

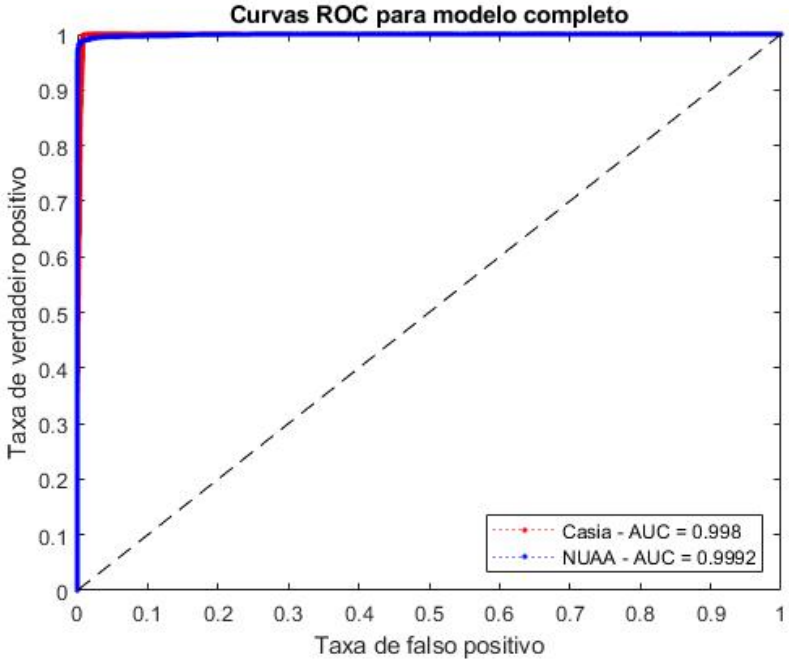
Conforme os resultados apresentados, nota-se que este modelo obtém excelente desempenho em ambos os *datasets*, como esperado, visto que foi treinado utilizando dados de ambos os conjuntos. Neste modelo destaca-se o desempenho no conjunto Casia, onde se obteve uma taxa de falsa rejeições de 0%, ou seja, o modelo não classificou erroneamente nenhum quadro genuíno. A partir das curvas ROC pode-se confirmar o desempenho do modelo, visto que se obteve uma área sob a curva muito próxima de 1.

De acordo com as métricas encontradas na literatura e apresentadas na Seção 4.3.2, outra maneira de avaliar o desempenho de sistemas de classificação é a utilização da curva DET. A Figura 33 apresenta estes gráficos para o modelo treinado com ambos os *datasets* quando avaliado no conjunto Casia e NUAA, de maneira similar a Figura 32.

A partir da curva DET, estima-se o limiar que resulta na taxa de falso positivo (FRR) igual a taxa de falso negativo (FAR) e calcula-se novamente as métricas de avaliação. A Tabela 12 apresenta os resultados encontrados quando $FAR \approx FRR$ visto que não foi encontrado pontos onde $FAR = FRR$. Além das métricas encontradas anteriormente, também é apresentado a *Equal Error Rate*.

Após obter estes resultados, o modelo utilizando o quadro inteiro foi então comparado com outros métodos presentes na literatura que são considerados referências na área de detecção de ataques de apresentação. Para tal, foi comparado o FAR, FRR e HTER entre o método proposto e os encontrados pela literatura. Estas comparações podem ser vistas nas Tabelas 13 e 14, que apresentam as comparações para o conjunto NUAA e Casia, respectivamente. Nota-se que valores representados como '-' são referentes a dados que não são apresentados no artigo original.

Figura 32: Curvas ROC para o modelo treinado com ambos os *datasets*. As linhas vermelha e azul representam o *dataset* Casia e NUAA, respectivamente.



Fonte: Autor

Tabela 12: Resultados encontrados após o ajuste para FAR = FRR.
Treinado Casia + NUAA

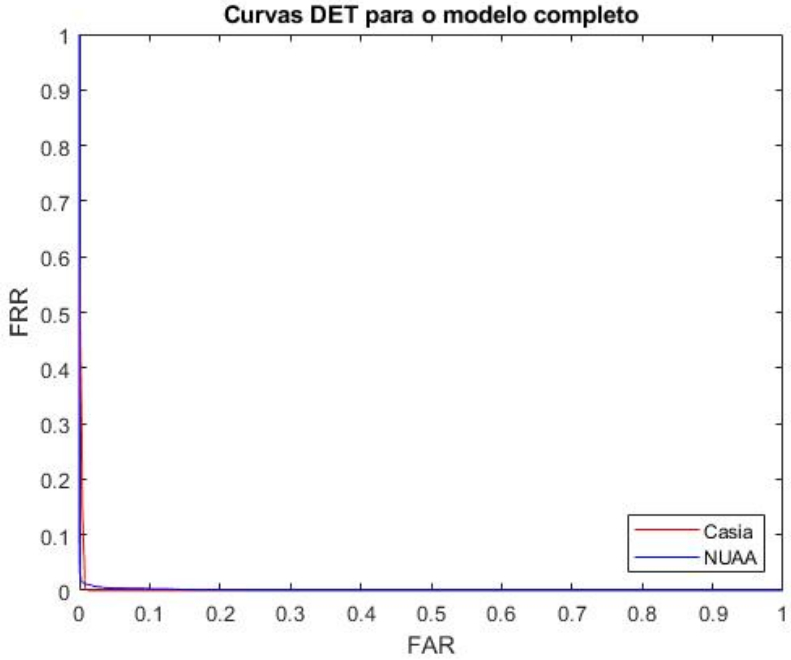
dataset testado	Limiar	CCR (%)	FAR (%)	FRR (%)	HTER (%)	EER (%)
NUAA	0.81	98.91	1.09	1.07	1.08	1.08
Casia	0.827	99.20	0.7996	0.7982	0.7989	0.8

Fonte: Autor

A partir das Tabelas 13 e 14, é possível notar que o modelo proposto atinge uma taxa de classificação correta média de 99%, o que representa um ganho de 22 e 4 pontos percentuais em relação aos outros métodos que também utilizam redes convolucionais. Dentre os fatores que contribuem para este ganho, se destaca o uso de informações de contexto da imagem, que se torna possível devido ao uso do quadro inteiro como entrada. Como mostrado nas Figuras 29, 30 e 31, ao realizar o treinamento contendo a informação completa capturada pela câmera, a rede aprender a utilizar informações do ambiente para realizar a discriminação entre as classes de ataque e tentativas de acesso genuínos. O uso de uma arquitetura mais complexa também é interessante, visto que ela é capaz de extrair feições de mais alto nível e por consequência tendem a gerar melhores resultados. A utilização desta rede mais profunda só foi possível graças ao emprego da técnica de transferência de aprendizado em conjunto com o aumento de dados, caso contrário, a base de dados CASIA e NUAA combinadas não iriam conter um número suficiente de amostrar para realizar o treinamento de tal rede, gerando um modelo com *overfitting*, que impactaria negativamente a taxa de classificação.

Além disso, como consequência da alta CCR, nota-se também bons resultados em termos tanto de FAR quanto de FRR, o que implica que o modelo está balanceado e não está fazendo classificações aleatórias baseado na classe com maior número de amostras (nota-se que o conjunto de testes não sofreu alterações, e portanto possui um número muito

Figura 33: Curvas DET para o modelo treinado em ambos os *datasets*. As linhas vermelha e azul representam o *dataset* Casia e NUAA, respectivamente.



Fonte: Autor

maior de amostras de impostores). Devido as baixas taxas de FAR e FRR, também se obteve um HTER baixo, de apenas 1.08% para o conjunto NUAA e 0.8% para o conjunto CASIA, batendo os outros métodos encontrados. Em adição, o método proposto também é mais simples que outros métodos que utilizam CNN visto que não é necessário uma etapa adicional de localização facial.

Ao considerar os resultados encontrados no conjunto NUAA, o modelo proposto obteve uma melhora de apenas 0.11%, que pode não justificar a criação de um novo modelo, no entanto, este conjunto possui dimensões reduzidas e já é datado. Porém quando o desempenho é avaliado no conjunto CASIA, nota-se uma situação mais favorável, onde foi possível reduzir o HTER praticamente pela metade, de 1.3% para 0.79% e obteve-se um ganho de quase 4% sobre o lsCNN e mais de 20% sobre o método com CNNs proposto por (AKBULUT et al., 2017), fazendo com que a proposta apresentada nesta dissertação tenha o melhor resultado utilizando redes neurais para o conjunto CASIA.

Tabela 13: Comparação entre o método proposto e outros presentes na literatura para o conjunto NUAA.

Resultados NUAA

Método	ACC(%)	FAR(%)	FRR(%)	HTER(%)
CNN (AKBULUT et al., 2017)	76.31	-	-	-
LRF-ELM (AKBULUT et al., 2017)	84.04	-	-	-
Uniform LBPV (KOSE; DUGELAY, 2012)	86.95	13.75	11.87	12.81
All LBPV (KOSE; DUGELAY, 2012)	88.03	13.61	9.16	11.38
Mult. LBP (MAATTA et al., 2011)	98.00	0.6	4.4	2.5
Qual. image (LUAN et al., 2017)	98.80	-	-	-
Proposto	**98.91**	**1.09**	**1.07**	**1.08**

Fonte: Autor

Tabela 14: Comparação entre o método proposto e outros presentes na literatura para o conjunto CASIA.

Resultados Casia

Método	ACC (%)	FAR (%)	FRR (%)	HTER (%)
CNN (AKBULUT et al., 2017)	76.31	-	-	-
LRF-ELM (AKBULUT et al., 2017)	84.04	-	-	-
LBP-TOP (FREITAS PEREIRA et al., 2013)	-	-	-	9.05
DoG+LBP (WEN; HAN; JAIN, 2015)	-	-	-	7.85
IDA+SVM (WEN; HAN; JAIN, 2015)	-	-	-	6.70
Deep Dictionary (MANJANI et al., 2017)	-	-	-	1.3
lsCNN (SOUZA; PAPA; MARANA, 2018)	95.56	-	-	-
Proposto	**99.20**	**0.7996**	**0.7982**	**0.7989**

Fonte: Autor

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi desenvolvido um modelo de rede neural convolucional para a detecção de ataques de apresentação, onde foi empregada uma nova maneira de treinar a rede, que consiste em utilizar o quadro inteiro capturado pela câmera ao invés de recortar apenas a face. Esta utilização se mostrou vantajosa visto que a rede neural tende a explorar feições tanto do ambiente, quanto do meio utilizado para realizar o ataque de apresentação para discriminar o quadro entre genuíno ou impostor, conforme apresentado nas Figuras 29, 30 e 31.

Além disto, também é proposto a utilização de transferência de aprendizado de redes neurais ao invés de construir um modelo do zero. Ao utilizar esta técnica, tira-se proveito do conhecimento obtido previamente pela rede quando ela foi treinada pela primeira vez. Por exemplo, a GoogLeNet quando treinada utilizando dados do ImageNet, aprendeu a extrair feições para realizar a classificação entre 1000 classes distintas, e essas feições podem ser ajustadas para detectar ataques de *spoofing*. Outras vantagens dessa abordagem são a diminuição do tempo de treinamento e redução do número de parâmetros a serem ajustados.

Conforme resultados apresentados no Capítulo 5, o modelo proposto apresenta um aumento da taxa de acertos de 95% quando utilizando somente faces para 99% ao utilizar a informação completa do quadro. Além de melhorar o desempenho em termos da métricas de avaliação, ao utilizar o quadro completo também se reduz a complexidade total do algoritmo visto que a etapa de localização facial não é mais necessária.

Futuramente, seria interessante a utilização de algum modelo que também utilize as informações da estrutura temporal dos dados, como uma rede neural recorrente. Desta forma seria possível capturar detalhes dos movimentos realizados pelo sujeito e assim aumentar o poder de discriminação do modelo. Ademais, a aquisição de uma placa gráfica reduziria consideravelmente o tempo de treinamento e teste dos modelos, tornando possível iterar mais rapidamente no projeto. Mesmo utilizando a técnica de transferência de aprendizado, este processo ainda leva cerca de 3h por época, o que acarreta um tempo total de 15h para o caso de 5 épocas, como utilizado neste trabalho. Além disso, durante a fase de testes, a CPU utilizada entrega somente 12 quadros por segundo, enquanto a GPU GT 750M, apesar de ultrapassada e ser de dispositivo móvel, alcança taxas próximas a 55 quadros por segundo. Com esse upgrade, também se tornaria factível realizar testes com outras arquiteturas de CNN que possuem mais camadas e uma menor taxa de erro no desafio ILSVRC, como a arquitetura VGG e ResNet. Outro ponto relevante seria a utilização de mais conjuntos de dados para treinar e avaliar o modelo, como o conjunto Replay-Attack e OULU.

REFERÊNCIAS

AHONEN, T.; HADID, A.; PIETIKAINEN, M. Face description with local binary patterns: application to face recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, New York, v.28, n.12, p.2037–2041, Dec. 2006.

AKBULUT, Y. et al. Deep learning based face liveness detection in videos. *In*: INTERNATIONAL ARTIFICIAL INTELLIGENCE AND DATA PROCESSING SYMPOSIUM (IDAP), 2., 2017, Malatya. **Proceedings...** New York: IEEE, 2017. p.1–4.

ALBU, R. D. Face anti-spoofing based on Radon transform. *In*: INTERNATIONAL CONFERENCE ON ENGINEERING OF MODERN ELECTRIC SYSTEMS (EMES), 13., 2015, Oradea. **Proceedings...** New York: IEEE, 2015. p.1–4.

ANJOS, A.; CHAKKA, M. M.; MARCEL, S. Motion-based counter-measures to photo attacks in face recognition. **IET Biometrics**, New York, v.3, n.3, p.147–158, Sept. 2014.

ANJOS, A.; MARCEL, S. Counter-measures to photo attacks in face recognition: a public database and a baseline. *In*: INTERNATIONAL JOINT CONFERENCE ON BIOMETRICS (IJCB), 2011, Washington, DC. **Proceedings...** New York: IEEE, 2011. n.1, p.1–7.

AYTAR, Y.; ZISSERMAN, A. Tabula rasa: model transfer for object category detection. *In*: INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2011, Barcelona. **Proceedings...** New York: IEEE, 2011. n.13, p.2252–2259.

BAO, W. et al. A liveness detection method for face recognition based on optical flow field. *In*: INTERNATIONAL CONFERENCE ON IMAGE ANALYSIS AND SIGNAL PROCESSING, 2009, Taizhou. **Proceedings...** New York: IEEE, 2009. n.15, p.233–236.

BENLAMOUDI, A. et al. Face spoofing detection using local binary patterns and Fisher Score. *In*: INTERNATIONAL CONFERENCE ON CONTROL, ENGINEERING INFORMATION TECHNOLOGY (CEIT), 3., 2015, Tlemcen. **Proceedings...** New York: IEEE, 2015. p.1–5.

BHARADWAJ, S. et al. Computationally Efficient Face Spoofing Detection with Motion Magnification. *In*: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION WORKSHOPS, 29., 2013, Portland. **Proceedings...** New York: IEEE, 2013. p.105–110.

BOULKENAFET, Z.; KOMULAINEN, J.; HADID, A. Face anti-spoofing based on color texture analysis. *In*: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2015, New York. **Proceedings...** IEEE, 2015. p.2636–2640.

COSTA-PAZO, A. et al. The Replay-Mobile face presentation-attack database. *In*: INTERNATIONAL CONFERENCE OF THE BIOMETRICS SPECIAL INTEREST GROUP (BIOSIG), 15., 2016, Darmstadt. **Proceedings...** New York: IEEE, 2016. p.1–7.

DUC, N. M.; MINH, B. Q. **Your face is not your password face authentication bypassing lenovo–asus–toshiba**., Black Hat Briefings, Las Vegas, 2009. Disponível em: <https://pdfs.semanticscholar.org/2eee/aba15fa6e88a221d5d7a4ce2014951a8a86e.pdf>. Acesso em: 27 Jun. 2019

FREITAS PEREIRA, T. de et al. LBP-TOP based countermeasure against face spoofing attacks. *In*: INTERNATIONAL CONFERENCE ON COMPUTER VISION - VOLUME PART I, 11., 2013, Berlin, Heidelberg. **Proceedings...** Berlin: Springer-Verlag, 2013. p.121–132.

FREITAS PEREIRA, T. de et al. Can face anti-spoofing countermeasures work in a real world scenario? *In*: INTERNATIONAL CONFERENCE ON BIOMETRICS (ICB), 6., 2013, Madrid. **Proceedings...** New York: IEEE, 2013. p.1–8.

GALBALLY, J.; MARCEL, S.; FIERREZ, J. Biometric antispoofing methods: a survey in face recognition. **IEEE Access**, New York, v.2, p.1530–1552, Dec. 2014.

GALBALLY, J.; MARCEL, S.; FIERREZ, J. Image quality assessment for fake biometric detection: application to iris, fingerprint, and face recognition. **IEEE Transactions on Image Processing**, New York, v.23, n.2, p.710–724, Feb. 2014.

Google. **Machine learning glossary**, 2019. Disponível em: <https://developers.google.com/machine-learning/glossary/>. Acesso em: 9 maio 2019.

HE, K. et al. Deep residual learning for image recognition. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 26., 2016. **Proceedings...** New York: IEEE, 2016. p.770–778.

HELD, D.; THRUN, S.; SAVARESE, S. Learning to track at 100 FPS with deep regression networks. *In*: EUROPEAN CONFERENCE IN COMPUTER VISION – ECCV, Cham. **Proceedings...** Springer International Publishing, 2016. p.749–765.

HUBEL, D.; WIESEL, T. Receptive fields and functional architecture of monkey striate cortex. **The Journal of Physiology**, Hoboken, v.195, p.215–243, Apr. 1968.

JAIN, A. K.; ROSS, A.; PRABHAKAR, S. An introduction to biometric recognition. **IEEE Transactions on Circuits and Systems for Video Technology**, New York, v.14, n.1, p.4–20, Jan. 2004.

KOLLREIDER, K. et al. Real-time face detection and motion analysis with application in liveness assessment. **IEEE Transactions on Information Forensics and Security**, New York, v.2, n.3, p.548–558, Sept. 2007.

KOMULAINEN, J. et al. Complementary countermeasures for detecting scenic face spoofing attacks. *In*: INTERNATIONAL CONFERENCE ON BIOMETRICS (ICB), 2013, Madrid. **Proceedings...** New York: IEEE, 2013. p.1–7.

KOMULAINEN, J.; HADID, A.; PIETIKAINEN, M. Context based face anti-spoofing. *In*: INTERNATIONAL CONFERENCE ON BIOMETRICS: THEORY, APPLICATIONS AND SYSTEMS (BTAS), 2013, Arlington, VA. **Proceedings...** New York: IEEE, 2013. p.1–8.

KOSE, N.; DUGELAY, J. L. Classification of captured and recaptured images to detect photograph spoofing. *In*: INTERNATIONAL CONFERENCE ON INFORMATICS, ELECTRONICS VISION (ICIEV), 2012, Dhaka. **Proceedings...** New York: IEEE, 2012. p.1027–1032.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet classification with deep convolutional neural networks. **Advances in neural information processing systems** . Lake Tahoe: Curran Associates, Inc., 2012. p.1097–1105.

LECUN, Y. et al. Object recognition with gradient-based learning. *In*: SHAPE, CONTOUR AND GROUPING IN COMPUTER VISION, 1999, Berlim. **Proceedings...** Berlim: Springer, 1999. p.319.

LI, J. et al. Live face detection based on the analysis of Fourier spectra. **Proc. SPIE**, Bellingham, v.5404, p.296–303, 2004.

LIU, S.; DENG, W. Very deep convolutional neural network based image classification using small training sample size. *In*: IAPR ASIAN CONFERENCE ON PATTERN RECOGNITION (ACPR), 2015, Kuala Lumpur. **Proceedings...** New York: IEEE, 2015. p.730–734.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2015, Boston, MA. **Proceedings...** New York: IEEE, 2015. p.3431–3440.

LUAN, X. et al. Face liveness detection with recaptured feature extraction. *In*: INTERNATIONAL CONFERENCE ON SECURITY, PATTERN ANALYSIS, AND CYBERNETICS (SPAC), 1., 2017, Shenzhen. **Proceedings...** New York: IEEE, 2017. p.429–432.

MAATTA, J.; HADID, A.; PIETIKAINEN, M. Face spoofing detection from single images using micro-texture analysis. *In*: INTERNATIONAL JOINT CONFERENCE ON BIOMETRICS (IJCB), 1., 2011, Washington. **Proceedings...** New York: IEEE, 2011. p.1–7.

MANJANI, I. et al. Detecting silicone mask-based presentation attack via deep dictionary learning. **Transactions on Information Forensics and Security**, New York, v.12, n.7, p.1713–1723, July 2017.

MARTIN, A. F. et al. The DET curve in assessment of detection task performance. *In*: EUROSPEECH, 2., 1997, Rhodes. **Proceedings...** Baixas: ISCA, 1997. p.22–25.

MATLAB r2018a. Natick, The MathWorks, 2018.

PAN, G. et al. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. *In*: INTERNATIONAL CONFERENCE ON COMPUTER VISION, 11., 2007, Rio de Janeiro. **Proceedings...** New York: IEEE, 2007. p.1–8.

PRATT, L. Y. Discriminability-based transfer between neural networks. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 1993, San Francisco. **Proceedings...** Burlington: Morgan Kaufmann Publishers Inc., 1993. p.204–211.

REDMON, J. et al. You only look once: unified, real-time object detection. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016, Las Vegas, NV. **Proceedings...** New York: IEEE, 2016. p.779–788.

ROTH, H. R. et al. DeepOrgan: multi-level deep convolutional networks for automated pancreas segmentation. *In*: MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION, 2015, Cham. **Proceedings...** Cham: Springer International Publishing, 2015. p.556–564.

SOLDERA, J. et al. Facial biometrics and applications. **IEEE Instrumentation Measurement Magazine**, New York, v.20, n.2, p.4–30, April 2017.

SOUZA, G. B. de et al. Deep texture features for robust face spoofing detection. **IEEE Transactions on Circuits and Systems II: Express Briefs**, New York, v.64, n.12, p.1397–1401, Dec. 2017.

SOUZA, G. B. de; PAPA, J. P.; MARANA, A. N. On the learning of deep local features for robust face spoofing detection. **Conference on Graphics, Patterns and Images**, 31., Foz do Iguaçu, v.31, p.258–265, 2018.

SOUZA, G. B. et al. Efficient transfer learning for robust face spoofing detection. *In*: PROGRESS IN PATTERN RECOGNITION, IMAGE ANALYSIS, COMPUTER VISION, AND APPLICATIONS, 2018, Cham. **Proceedings...** Cham: Springer International Publishing, 2018. p.643–651.

SOUZA, L. et al. How far did we get in face spoofing detection? **Eng. Appl. Artif. Intell.**, Oxford, v.72, n.C, p.368–381, 2018.

SPRINGER. **Enciclopédia Springer de Machine Learning**, 2011. Disponível em: <https://link.springer.com/referenceworkentry/10.1007\%2F978-0-387-30164-8_401>. Acesso em: 28 fev. 2019.

SUN, X.; HUANG, L.; LIU, C. Context based face spoofing detection using active near-infrared images. *In*: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION (ICPR), 2016, Cancun. **Proceedings...** New York: IEEE, 2016. p.4262–4267.

SZEGEDY, C. et al. Going deeper with convolutions. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2015, Boston, MA. **Proceedings...** New York: IEEE, 2015. p.1–9.

TAN, X. et al. Face liveness detection from a single image with sparse low rank bilinear discriminative model. *In*: COMPUTER VISION – ECCV, 2010, Berlin, Heidelberg. **Proceedings...** Berlin: Springer, 2010. p.504–517.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. *In*: COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2001, Kauai, HI. **Proceedings...** New York: IEEE, 2001. v.1, p.I–511–I–518.

WANG, T. et al. Face liveness detection using 3D structure recovered from a single camera. *In*: INTERNATIONAL CONFERENCE ON BIOMETRICS (ICB), 2013, Madrid. **Proceedings...** New York: IEEE, 2013. p.1–6.

WEN, D.; HAN, H.; JAIN, A. K. Face ppoof detection with image distortion analysis. **IEEE Transactions on Information Forensics and Security**, New York, v.10, n.4, p.746–761, April 2015.

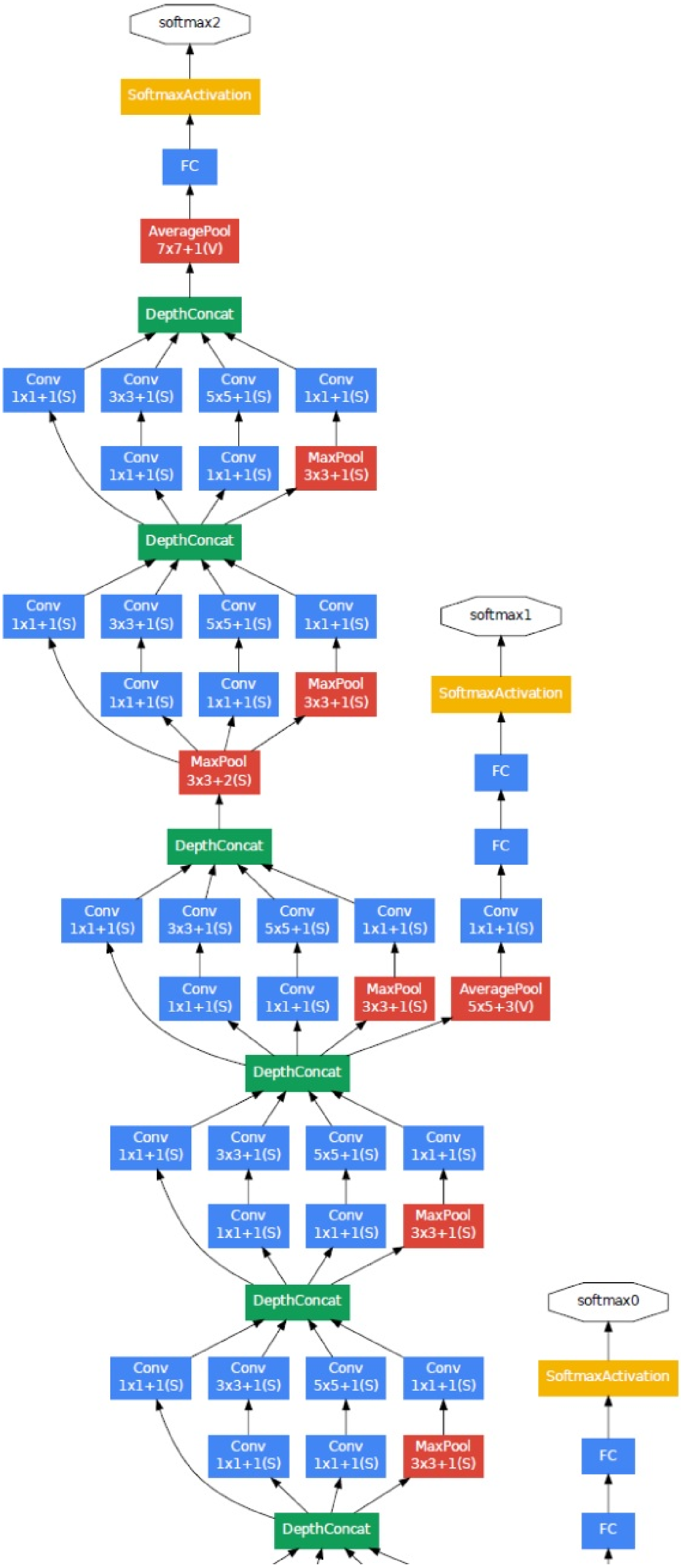
ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. *In*: EUROPEN CONFERENCE ON COMPUTER VISION, 2014, Cham. **Proceedings...** Cham: Springer International Publishing, 2014. p.818–833.

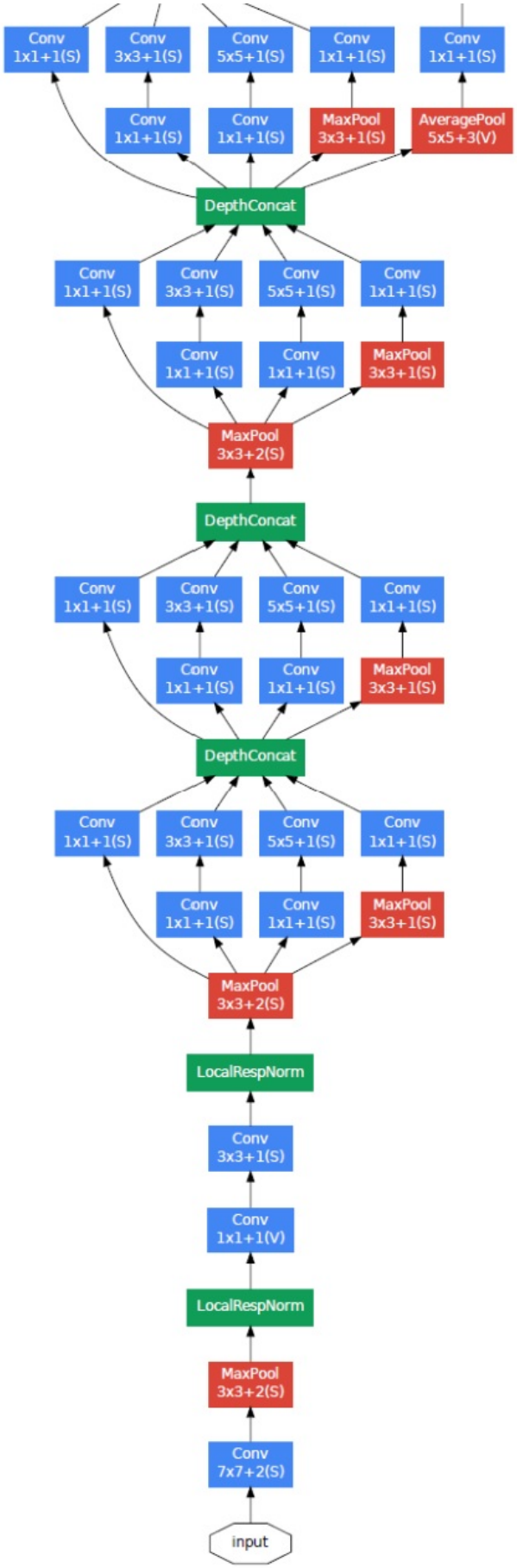
ZHANG, Z. et al. A face antispoofing database with diverse attacks. *In*: INTERNATIONAL CONFERENCE ON BIOMETRICS (ICB), 5., 2012, New Delhi. **Proceedings...** New York: IEEE, 2012. p.26–31.

ANEXO A GOOGLENET COMPLETA

A Figura A apresenta a arquiteta completa da GoogLeNet. Devido a grande quantidade de camadas presentes nesta rede, a figura teve que ser divida em duas páginas completa para facilitar sua leitura.

Figura 34: Arquitetura GoogLeNet completa.





Fonte: (SZEGEDY et al., 2015)